Practical 3

Aim: Process Management

1. Managing and monitoring linux processes
2. Control Services and Daemons
3. Improve Command Line productivity

Commands for reference:

Process: top, ps, kill, pkill, w, lscpu

Control Services and Daemons: systemctl with parameters start, stop, restart, enabledisable, is-active, is-enabled and is-failed service

I/O Redirection (<, >, >>), Pipe (|)

Part-A

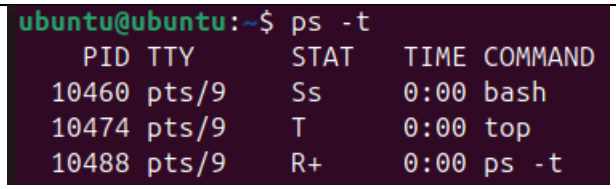
Monitoring and Managing Linux Processes

Que.	**1. List down all processes with their states sorted by their CPU Usage. Identify current running process.**
Command	ps

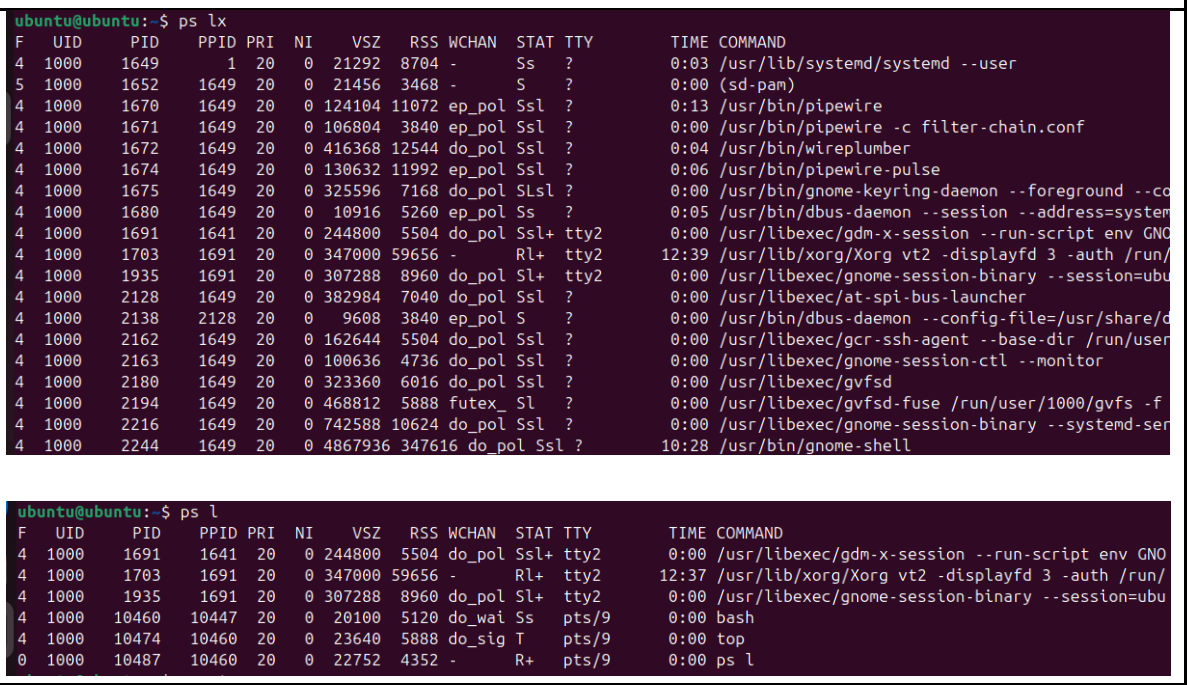
Output	`ubuntu@ubuntu:~$ ps`
	`PID TTY TIME CMD`
	`10460 pts/9 00:00:00 bash`
	`10474 pts/9 00:00:00 top`
	`10477 pts/9 00:00:00 ps`

Que.	**2. List down all processes associated with current user.**
Command	Ps -u

Output	`ubuntu@ubuntu:~$ ps -u`
	`USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND`
	`ubuntu 1691 0.0 0.2 244800 5504 tty2 Ssl+ 01:55 0:00 /usr/libexec/gdm-x-session --run-script env GNOME_SHE`
	`ubuntu 1703 5.3 2.9 347000 59656 tty2 Rl+ 01:55 12:38 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1`
	`ubuntu 1935 0.0 0.4 307288 8960 tty2 Sl+ 01:55 0:00 /usr/libexec/gnome-session-binary --session=ubuntu`
	`ubuntu 10460 0.0 0.2 20100 5120 pts/9 Ss 05:44 0:00 bash`
	`ubuntu 10474 0.0 0.2 23640 5888 pts/9 T 05:45 0:00 top`
	`ubuntu 10489 200 0.2 22720 4480 pts/9 R+ 05:53 0:00 ps -u`

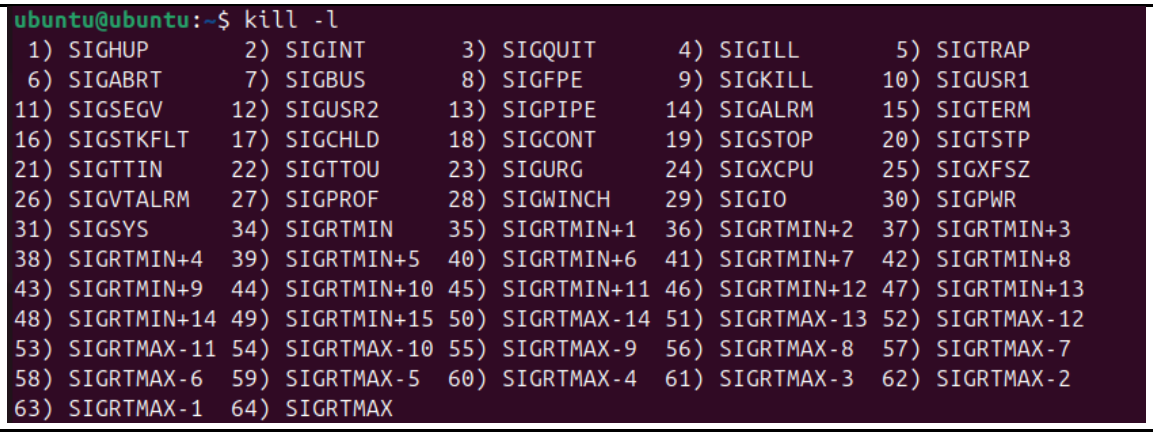
Que.	3. List down all processes associated with their terminal and their states. Identify current running process.
Command	Ps -t
Output	

```
ubuntu@ubuntu:~$ ps -t
    PID TTY      STAT   TIME COMMAND
  10460 pts/9    Ss     0:00 bash
  10474 pts/9    T      0:00 top
  10488 pts/9    R+     0:00 ps -t
```

Que.	4. Compare the output of “ps lx” and “ps l” commands
Command	Ps lx Ps l
Output	

```
ubuntu@ubuntu:~$ ps lx
F   UID     PID    PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4  1000    1649       1  20   0  21292  8704 -      Ss   ?          0:03 /usr/lib/systemd/systemd --user
5  1000    1652    1649  20   0  21456  3468 -      S    ?          0:00 (sd-pam)
4  1000    1670    1649  20   0 124104 11072 ep_pol Ssl  ?          0:13 /usr/bin/pipewire
4  1000    1671    1649  20   0 106804  3840 ep_pol Ssl  ?          0:00 /usr/bin/pipewire -c filter-chain.conf
4  1000    1672    1649  20   0 416368 12544 do_pol Ssl  ?          0:04 /usr/bin/wireplumber
4  1000    1674    1649  20   0 130632 11992 ep_pol Ssl  ?          0:06 /usr/bin/pipewire-pulse
4  1000    1675    1649  20   0 325596  7168 do_pol SLsl ?          0:00 /usr/bin/gnome-keyring-daemon --foreground --co
4  1000    1680    1649  20   0  10916  5260 ep_pol Ss   ?          0:05 /usr/bin/dbus-daemon --session --address=system
4  1000    1691    1641  20   0 244800  5504 do_pol Ssl+ tty2       0:00 /usr/libexec/gdm-x-session --run-script env GNO
4  1000    1703    1691  20   0 347000 59656 -      Rl+  tty2      12:39 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/
4  1000    1935    1691  20   0 307288  8960 do_pol Sl+  tty2       0:00 /usr/libexec/gnome-session-binary --session=ubu
4  1000    2128    1649  20   0 382984  7040 do_pol Ssl  ?          0:00 /usr/libexec/at-spi-bus-launcher
4  1000    2138    2128  20   0   9608  3840 ep_pol S    ?          0:00 /usr/bin/dbus-daemon --config-file=/usr/share/d
4  1000    2162    1649  20   0 162644  5504 do_pol Ssl  ?          0:00 /usr/libexec/gcr-ssh-agent --base-dir /run/user
4  1000    2163    1649  20   0 100636  4736 do_pol Ssl  ?          0:00 /usr/libexec/gnome-session-ctl --monitor
4  1000    2180    1649  20   0 323360  6016 do_pol Ssl  ?          0:00 /usr/libexec/gvfsd
4  1000    2194    1649  20   0 468812  5888 futex_ Sl   ?          0:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f
4  1000    2216    1649  20   0 742588 10624 do_pol Ssl  ?          0:00 /usr/libexec/gnome-session-binary --systemd-ser
4  1000    2244    1649  20   0 4867936 347616 do_pol Ssl ?        10:28 /usr/bin/gnome-shell
```

```
ubuntu@ubuntu:~$ ps l
F   UID     PID    PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4  1000    1691    1641  20   0 244800  5504 do_pol Ssl+ tty2       0:00 /usr/libexec/gdm-x-session --run-script env GNO
4  1000    1703    1691  20   0 347000 59656 -      Rl+  tty2      12:37 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/
4  1000    1935    1691  20   0 307288  8960 do_pol Sl+  tty2       0:00 /usr/libexec/gnome-session-binary --session=ubu
4  1000   10460   10447  20   0  20100  5120 do_wai Ss   pts/9      0:00 bash
4  1000   10474   10460  20   0  23640  5888 do_sig T    pts/9      0:00 top
0  1000   10487   10460  20   0  22752  4352 -      R+   pts/9      0:00 ps l
```

Que.	5. List down all the names and numbers of all available signals
Command	kill -l
Output	

```
ubuntu@ubuntu:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Que.	6. Run the “sleep 10000” in background.
Command	sleep 10000 &
Output	ubuntu@ubuntu:~$ sleep 10000 & [2] 10594

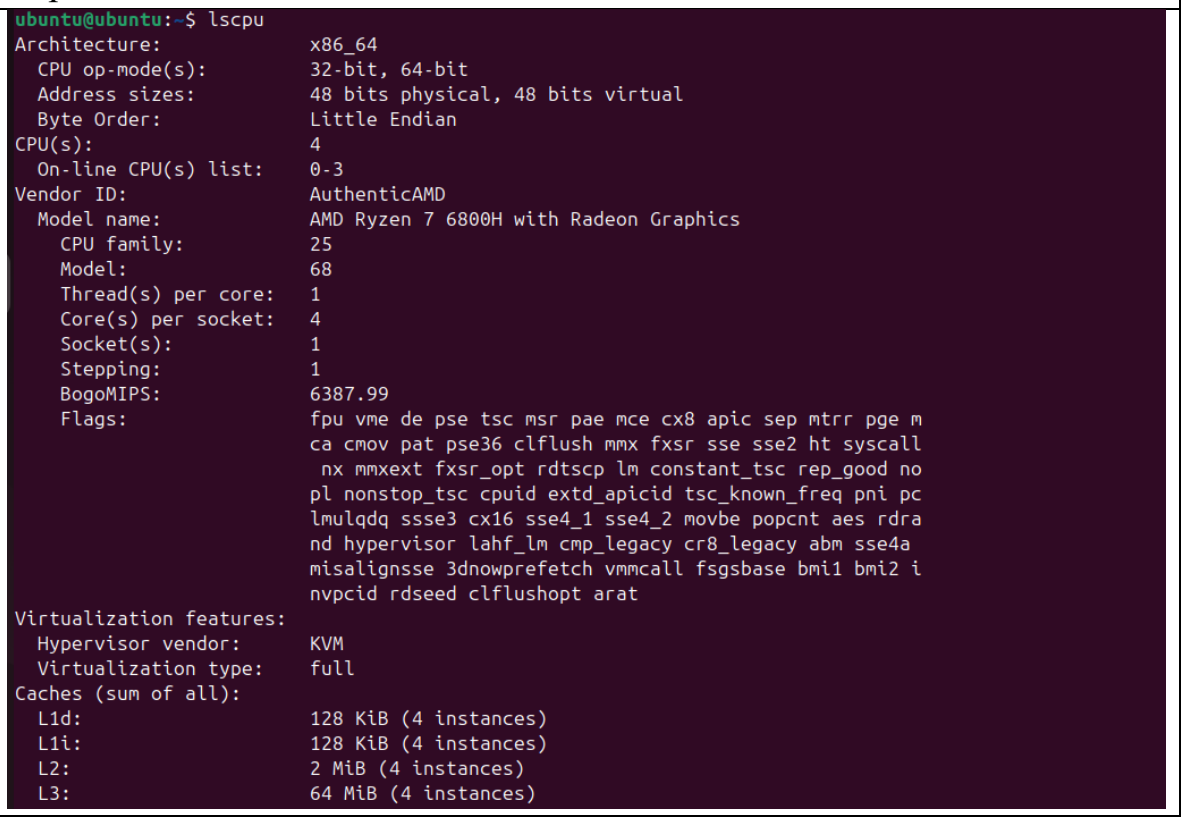
Que.	7.Check the PID of sleep process and kill it using PID.
Command	pgrep sleep kill (PID)
Output	ubuntu@ubuntu:~$ pgrep sleep 10594 ubuntu@ubuntu:~$ kill 10594

Que.	8.Apply w command and observer the output
Command	w
Output	ubuntu@ubuntu:~$ w 05:23:12 up 4:07, 1 user, load average: 0.05, 0.09, 0.12 USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT ubuntu tty2 - 22Jul24 13days 12:45 0.21s /usr/libexec/gnome-session-binary --session=ubuntu [2]- Terminated sleep 10000

Que.	9. Open the firefox browser. Check the processes associated with firefox
Command	top
Output	ubuntu@ubuntu:~$ top top - 05:26:26 up 4:10, 1 user, load average: 1.10, 0.61, 0.31 Tasks: 253 total, 2 running, 243 sleeping, 8 stopped, 0 zombie %Cpu(s): 7.8 us, 8.6 sy, 0.0 ni, 83.3 id, 0.0 wa, 0.0 hi, 0.3 si, 0.0 st MiB Mem : 1967.2 total, 82.4 free, 1774.0 used, 508.9 buff/cache MiB Swap: 0.0 total, 0.0 free, 0.0 used. 193.2 avail Mem PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 2244 ubuntu 20 0 4871596 315980 39304 R 75.5 15.7 11:52.76 gnome-shell 1703 ubuntu 20 0 378732 88196 37004 S 4.6 4.4 12:57.11 Xorg 10599 ubuntu 20 0 2938092 265500 129120 S 0.7 13.2 0:07.71 firefox 11422 ubuntu 20 0 23616 5632 3456 R 0.7 0.3 0:00.07 top

Que.	10. Kill all processes associated with firefox by its name.
Command	Pkill firefox
Output	ubuntu@ubuntu:~$ pkill firefox

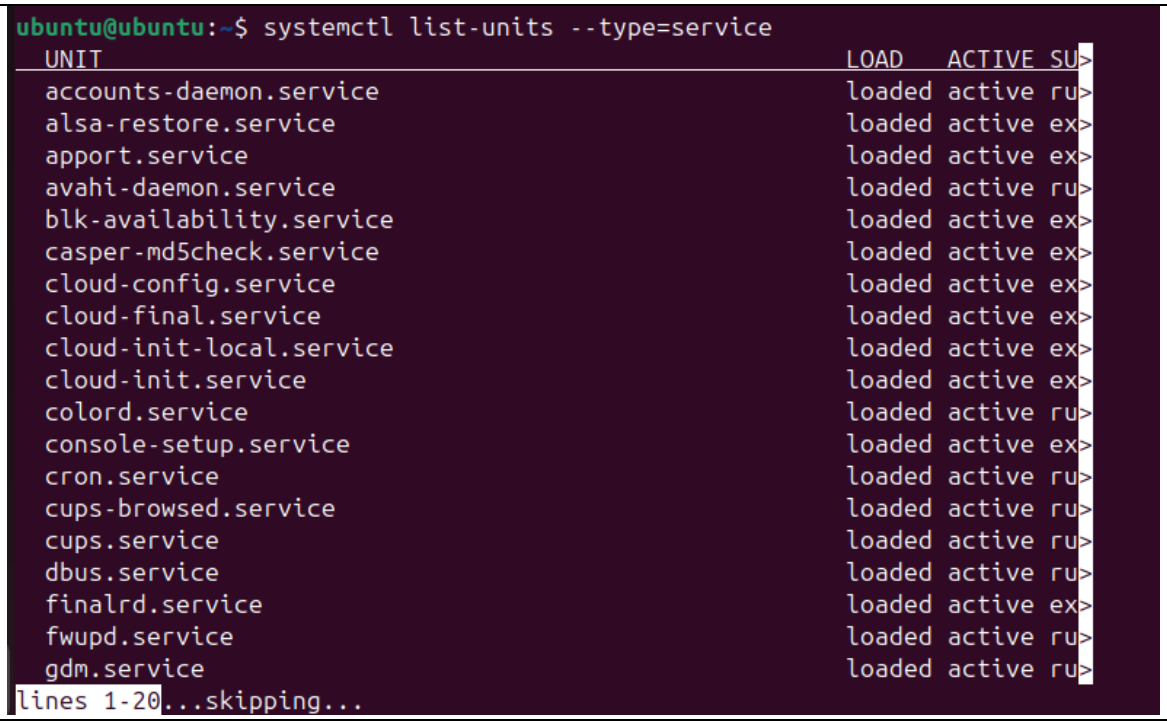
Que.	11. Give the difference between kill and pkill.
Output	kill: Sends a signal to a process specified by its PID. pkill: Sends a signal to all processes that match a specified name or other attributes.

Que.	12. Run “lscpu” command and observer the output.
Command	lscpu
Output	

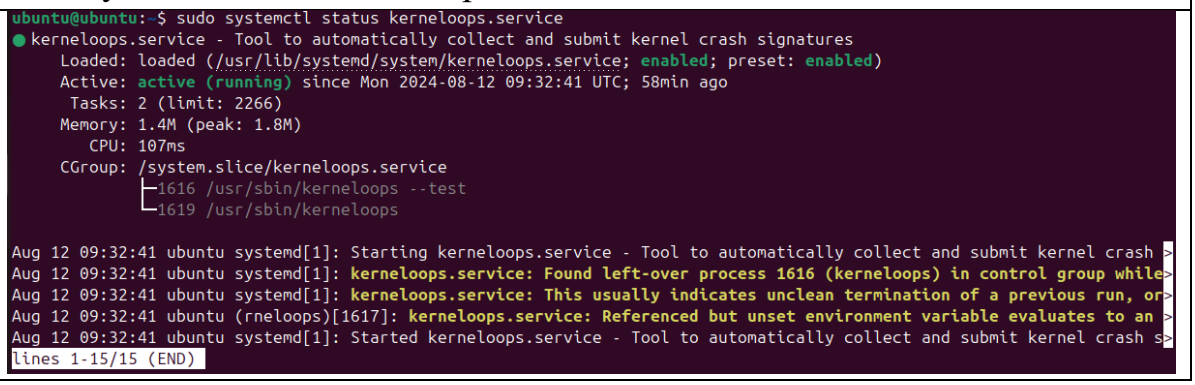
```
ubuntu@ubuntu:~$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          48 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   4
  On-line CPU(s) list:    0-3
Vendor ID:                AuthenticAMD
  Model name:             AMD Ryzen 7 6800H with Radeon Graphics
    CPU family:           25
    Model:                68
    Thread(s) per core:   1
    Core(s) per socket:   4
    Socket(s):            1
    Stepping:             1
    BogoMIPS:             6387.99
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge m
                          ca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall
                           nx mmxext fxsr_opt rdtscp lm constant_tsc rep_good no
                          pl nonstop_tsc cpuid extd_apicid tsc_known_freq pni pc
                          lmulqdq ssse3 cx16 sse4_1 sse4_2 movbe popcnt aes rdra
                          nd hypervisor lahf_lm cmp_legacy cr8_legacy abm sse4a
                          misalignsse 3dnowprefetch vmmcall fsgsbase bmi1 bmi2 i
                          nvpcid rdseed clflushopt arat
Virtualization features:
  Hypervisor vendor:      KVM
  Virtualization type:    full
Caches (sum of all):
  L1d:                    128 KiB (4 instances)
  L1i:                    128 KiB (4 instances)
  L2:                     2 MiB (4 instances)
  L3:                     64 MiB (4 instances)
```

PART B

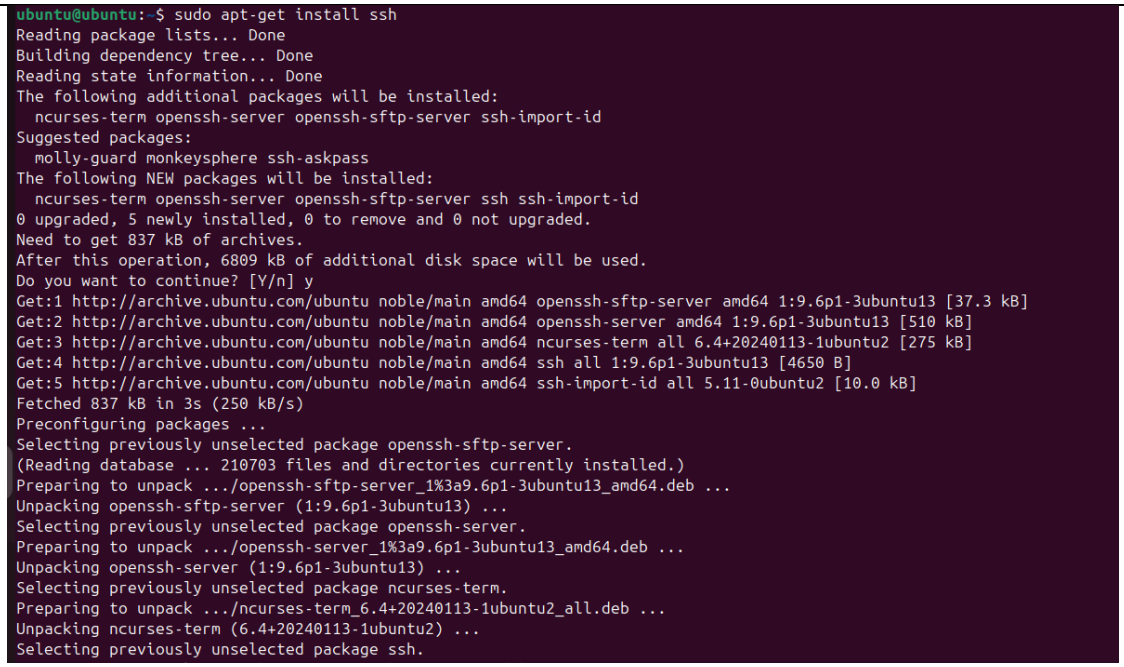
Control Services and daemons

Que.	1. List all services on your system.(systemctl list-units --type=service)
Command	systemctl list-units --type=service
Output	

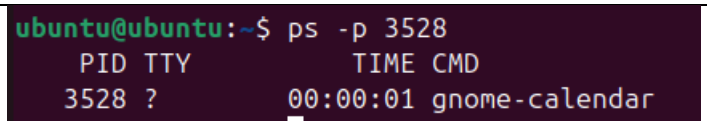
```
ubuntu@ubuntu:~$ systemctl list-units --type=service
  UNIT                                                LOAD   ACTIVE SU>
  accounts-daemon.service                             loaded active ru>
  alsa-restore.service                                loaded active ex>
  apport.service                                      loaded active ex>
  avahi-daemon.service                                loaded active ru>
  blk-availability.service                            loaded active ex>
  casper-md5check.service                             loaded active ex>
  cloud-config.service                                loaded active ex>
  cloud-final.service                                 loaded active ex>
  cloud-init-local.service                            loaded active ex>
  cloud-init.service                                  loaded active ex>
  colord.service                                      loaded active ru>
  console-setup.service                               loaded active ex>
  cron.service                                        loaded active ru>
  cups-browsed.service                                loaded active ru>
  cups.service                                        loaded active ru>
  dbus.service                                        loaded active ru>
  finalrd.service                                     loaded active ex>
  fwupd.service                                       loaded active ru>
  gdm.service                                         loaded active ru>
lines 1-20...skipping...
```

Que.	2. Check whether the ssh service is active or not. (sudosystemctl status service_name)
Command	sudo systemctl status kerneloops.service
Output	

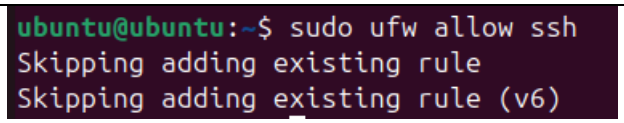
```
ubuntu@ubuntu:~$ sudo systemctl status kerneloops.service
● kerneloops.service - Tool to automatically collect and submit kernel crash signatures
     Loaded: loaded (/usr/lib/systemd/system/kerneloops.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-08-12 09:32:41 UTC; 58min ago
      Tasks: 2 (limit: 2266)
     Memory: 1.4M (peak: 1.8M)
        CPU: 107ms
     CGroup: /system.slice/kerneloops.service
             ├─1616 /usr/sbin/kerneloops --test
             └─1619 /usr/sbin/kerneloops

Aug 12 09:32:41 ubuntu systemd[1]: Starting kerneloops.service - Tool to automatically collect and submit kernel crash >
Aug 12 09:32:41 ubuntu systemd[1]: kerneloops.service: Found left-over process 1616 (kerneloops) in control group while>
Aug 12 09:32:41 ubuntu systemd[1]: kerneloops.service: This usually indicates unclean termination of a previous run, or>
Aug 12 09:32:41 ubuntu (rneloops)[1617]: kerneloops.service: Referenced but unset environment variable evaluates to an >
Aug 12 09:32:41 ubuntu systemd[1]: Started kerneloops.service - Tool to automatically collect and submit kernel crash s>
lines 1-15/15 (END)
```

Que.	3. If the package is not available, install ssh package (sudo apt-get install ssh)
Command	sudo apt-get install ssh
Output	

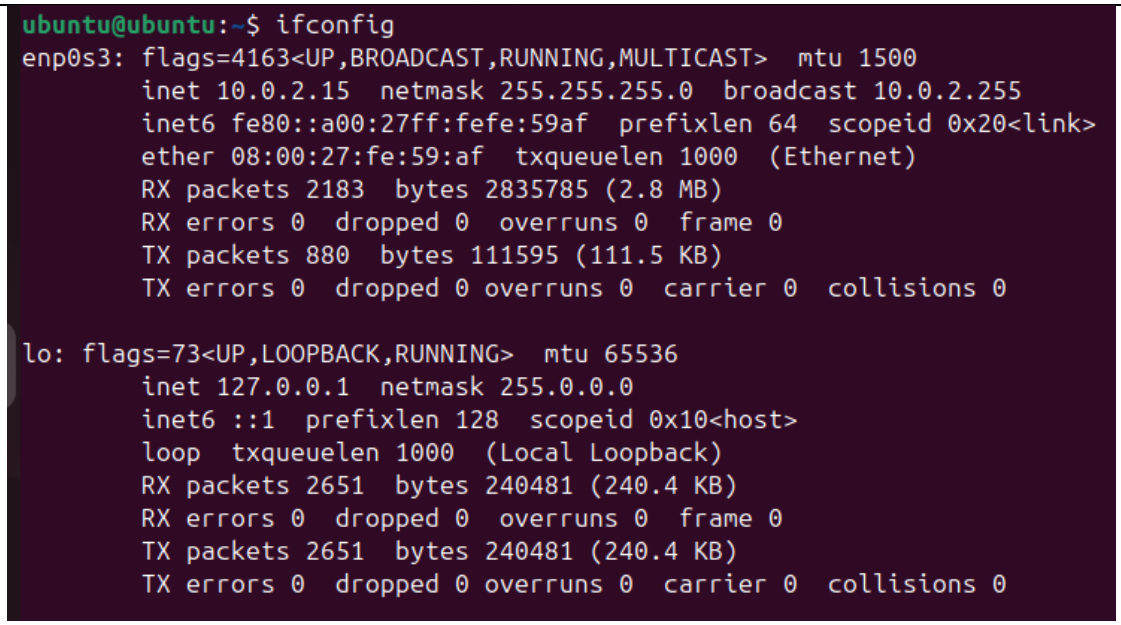
```
ubuntu@ubuntu:~$ sudo apt-get install ssh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh ssh-import-id
0 upgraded, 5 newly installed, 0 to remove and 0 not upgraded.
Need to get 837 kB of archives.
After this operation, 6809 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 openssh-sftp-server amd64 1:9.6p1-3ubuntu13 [37.3 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/main amd64 openssh-server amd64 1:9.6p1-3ubuntu13 [510 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/main amd64 ncurses-term all 6.4+20240113-1ubuntu2 [275 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/main amd64 ssh all 1:9.6p1-3ubuntu13 [4650 B]
Get:5 http://archive.ubuntu.com/ubuntu noble/main amd64 ssh-import-id all 5.11-0ubuntu2 [10.0 kB]
Fetched 837 kB in 3s (250 kB/s)
Preconfiguring packages ...
Selecting previously unselected package openssh-sftp-server.
(Reading database ... 210703 files and directories currently installed.)
Preparing to unpack .../openssh-sftp-server_1%3a9.6p1-3ubuntu13_amd64.deb ...
Unpacking openssh-sftp-server (1:9.6p1-3ubuntu13) ...
Selecting previously unselected package openssh-server.
Preparing to unpack .../openssh-server_1%3a9.6p1-3ubuntu13_amd64.deb ...
Unpacking openssh-server (1:9.6p1-3ubuntu13) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../ncurses-term_6.4+20240113-1ubuntu2_all.deb ...
Unpacking ncurses-term (6.4+20240113-1ubuntu2) ...
Selecting previously unselected package ssh.
```

Que.	4. If the service is available and active, check the process state usng ps –p PID
Command	ps -p 3528
Output	

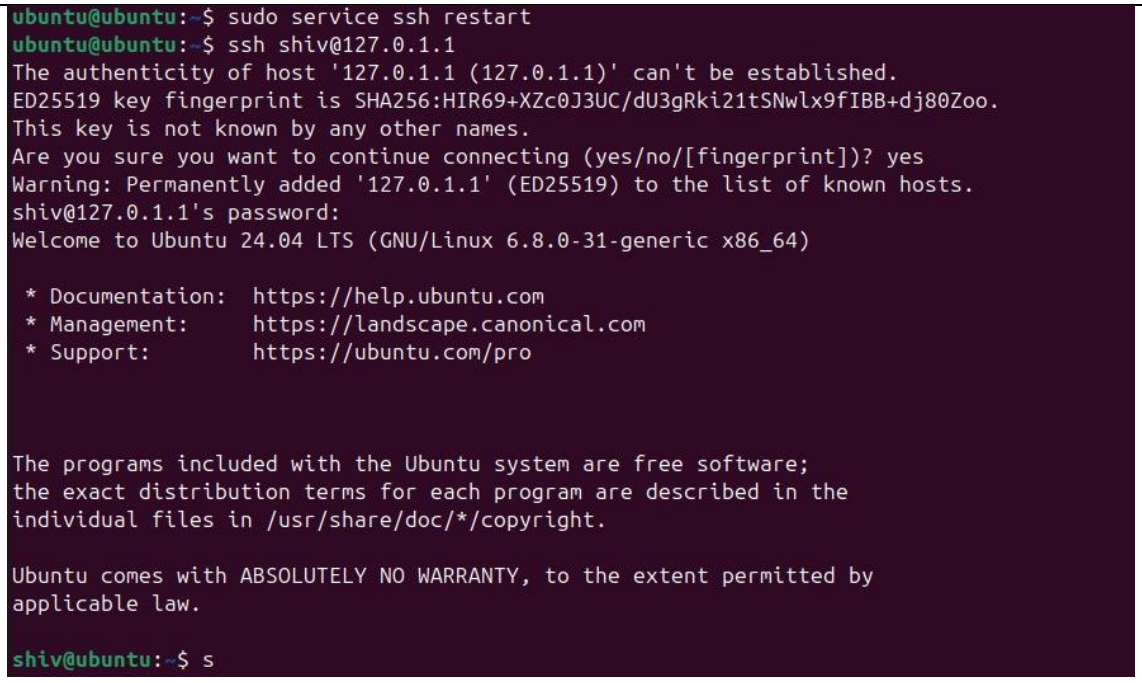
```
ubuntu@ubuntu:~$ ps -p 3528
    PID TTY          TIME CMD
   3528 ?        00:00:01 gnome-calendar
```

Que.	5. Add the firewall rule to allow remote service using ssh(sudo ufw allow ssh)
Command	sudo ufw allow ssh
Output	

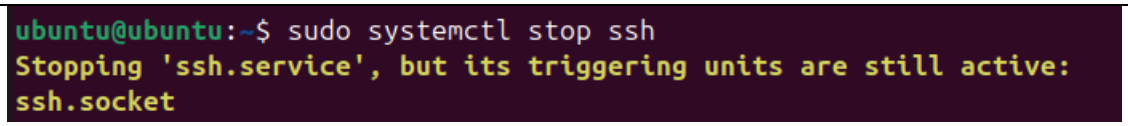
```
ubuntu@ubuntu:~$ sudo ufw allow ssh
Skipping adding existing rule
Skipping adding existing rule (v6)
```

Que.	6. Check your IP address
Command	ifconfig
Output	

```
ubuntu@ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fefe:59af  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:fe:59:af  txqueuelen 1000  (Ethernet)
        RX packets 2183  bytes 2835785 (2.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 880  bytes 111595 (111.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2651  bytes 240481 (240.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2651  bytes 240481 (240.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Que.	7. Access another user terminal using ssh
Command	ssh shiv@127.0.1.1
Output	

```
ubuntu@ubuntu:~$ sudo service ssh restart
ubuntu@ubuntu:~$ ssh shiv@127.0.1.1
The authenticity of host '127.0.1.1 (127.0.1.1)' can't be established.
ED25519 key fingerprint is SHA256:HIR69+XZc0J3UC/dU3gRki21tSNwlx9fIBB+dj80Zoo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.1.1' (ED25519) to the list of known hosts.
shiv@127.0.1.1's password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-31-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

shiv@ubuntu:~$ s
```

Que.	8. Stop the service and check the status
Command	sudo systemctl stop ssh
Output	

```
ubuntu@ubuntu:~$ sudo systemctl stop ssh
Stopping 'ssh.service', but its triggering units are still active:
ssh.socket
```

Que.	**9. Disable the service and check the status**
Command	sudo systemctl disable ssh

Output	`ubuntu@ubuntu:~$ sudo systemctl disable ssh`
	`Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.`
	`Executing: /usr/lib/systemd/systemd-sysv-install disable ssh`
	`Disabling 'ssh.service', but its triggering units are still active:`
	`ssh.socket`

Que.	**10. Enable it again and check the status**
Command	sudo systemctl enable ssh

Output	`ubuntu@ubuntu:~$ sudo systemctl enable ssh`
	`Synchronizing state of ssh.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.`
	`Executing: /usr/lib/systemd/systemd-sysv-install enable ssh`
	`Created symlink /etc/systemd/system/sshd.service → /usr/lib/systemd/system/ssh.service.`
	`Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service → /usr/lib/systemd/system/ssh.service.`

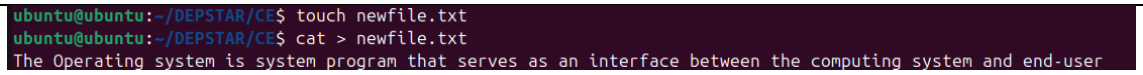
Que.	**11. Restart the service and check the status**
Command	sudo systemctl restart ssh
Output	`ubuntu@ubuntu:~$ sudo systemctl restart ssh`

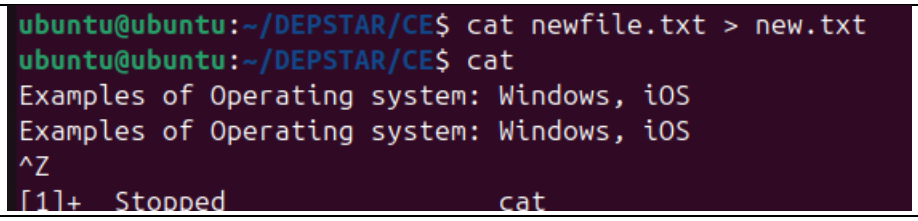
Que.	**12. Observe the analyze the output of be low mentioned command** **1. systemctl is-active ssh** **2. systemctl is-enabled ssh** **3. systemctl is- failed ssh**
Command	1. systemctl is-active ssh 2. systemctl is-enabled ssh 3. systemctl is- failed ssh

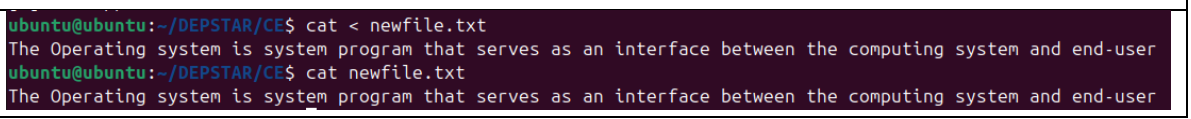
Output	`ubuntu@ubuntu:~$ systemctl is-active ssh`
	`active`
	`ubuntu@ubuntu:~$ systemctl is-enabled ssh`
	`enabled`
	`ubuntu@ubuntu:~$ systemctl is-failed ssh`
	`active`

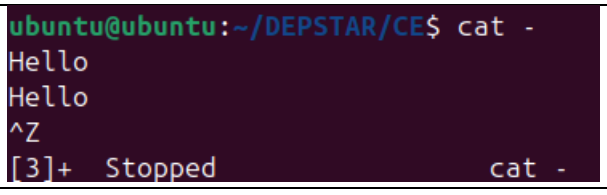
PART C

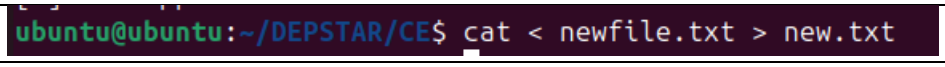
Improve Command Line Productivity
I/o Redirection

Que.	1. Create a file named “newfile.txt” and insert a text into created file as follow: “The operating system is a system program that serves as an interface between the computing system and the end-user.”
Command	touch newfile.txt Cat > newfile.txt
Output	

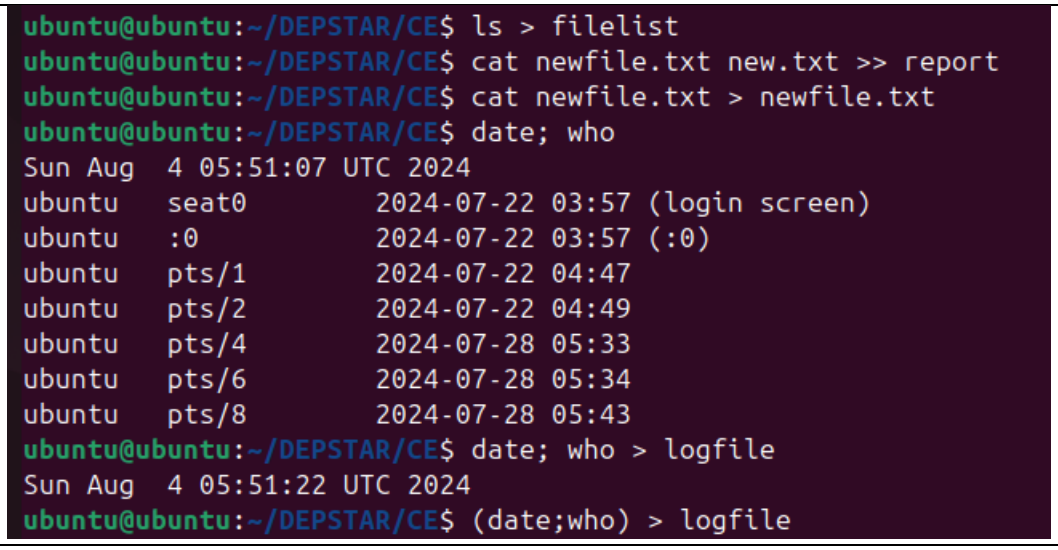
Que.	2. Redirect the output of “newfile.txt” file to file “new.txt” using command. 3. Type command cat, then enter key and enter some text. Observe the output
Command	Cat newfile.txt > new.txt cat
Output	

Que.	4. Type command i) cat <newfile.txt ii) cat newfile.txt. Interpret the output in both cases.
Command	i. cat < newfile.txt ii. cat newfile.txt
Output	

Que.	5. Type command cat – and enter any text.
Command	cat -
Output	

Que.	6. Type command i) cat < and > at once to redirect the output of one file to another.
Command	cat < newfile.txt > new.txt
Output	

Que.	**7. Summarize the use of cat command with redirection operator based on your done exercise.**
Output	cat file: Displays the contents of file. cat < file: Reads from file and outputs the contents. cat > file: Takes input from the keyboard and writes it to file. Press Ctrl+D to end the input. cat file1 > file2: Redirects the contents of file1 to file2, overwriting file2. cat file1 >> file2: Appends the contents of file1 to file2. cat -: Reads from standard input and echoes the input back.

Que.	**8. Try following command and interpret the output:** **a. ls >filelist** **b. cat newfile.txt new.txt >> report** **c. cat newfile.txt > newfile.txt** **d. date; who** **e. date; who>logfile** **f. (date; who) > logfile**
Command	a. ls >filelist b. cat newfile.txt new.txt >> report c. cat newfile.txt > newfile.txt d. date; who e. date; who>logfile f. (date; who) > logfile
Output	

```
ubuntu@ubuntu:~/DEPSTAR/CE$ ls > filelist
ubuntu@ubuntu:~/DEPSTAR/CE$ cat newfile.txt new.txt >> report
ubuntu@ubuntu:~/DEPSTAR/CE$ cat newfile.txt > newfile.txt
ubuntu@ubuntu:~/DEPSTAR/CE$ date; who
Sun Aug  4 05:51:07 UTC 2024
ubuntu   seat0        2024-07-22 03:57 (login screen)
ubuntu   :0           2024-07-22 03:57 (:0)
ubuntu   pts/1        2024-07-22 04:47
ubuntu   pts/2        2024-07-22 04:49
ubuntu   pts/4        2024-07-28 05:33
ubuntu   pts/6        2024-07-28 05:34
ubuntu   pts/8        2024-07-28 05:43
ubuntu@ubuntu:~/DEPSTAR/CE$ date; who > logfile
Sun Aug  4 05:51:22 UTC 2024
ubuntu@ubuntu:~/DEPSTAR/CE$ (date;who) > logfile
```

Piping

Que.	**1. ls | wc -l**
Command	ls | wc -l
Output	`ubuntu@ubuntu:~/DEPSTAR/CE$ ls | wc -l` `17`

Que.	**2.ls | less**
Command	ls | less

Output	`22DCE082_20240727.txt`
	`ce.txt`
	`f3.txt`
	`file1.txt`
	`file2.txt`
	`file4.txt`
	`file5.txt`
	`file6.txt`
	`file7.txt`
	`file8.txt`
	`file9.txt`
	`filelist`
	`logfile`
	`new.txt`
	`newfile.txt`
	`no_digit.txt`
	`report`
	`(END)`

Que.	**3. store the value of count in file named "countfile" using pipeline.** **4. Try command who | sort and observe the output.**
Command	Ls | wc – l > countfile who | sort

Output	`ubuntu@ubuntu:~/DEPSTAR/CE$ ls | wc -l > countfile`
	`ubuntu@ubuntu:~/DEPSTAR/CE$ who | sort`
	`ubuntu :0 2024-07-22 03:57 (:0)`
	`ubuntu pts/1 2024-07-22 04:47`
	`ubuntu pts/2 2024-07-22 04:49`
	`ubuntu pts/4 2024-07-28 05:33`
	`ubuntu pts/6 2024-07-28 05:34`
	`ubuntu pts/8 2024-07-28 05:43`
	`ubuntu seat0 2024-07-22 03:57 (login screen)`

Que.	**5. Store the sorted output in file named "sortedlist"**
Command	Sort newfile.txt > new.txt
Output	`ubuntu@ubuntu:~/DEPSTAR/CE$ sort newfile.txt > new.txt`

Que.	**6. Try cal 1996 | head -10**
Command	cal 1996 | head-10

Output	`ubuntu@ubuntu:~$ cal 1996 | head-10`
	`Command 'cal' not found, but can be installed with:`
	`sudo apt install ncal`
	`head-10: command not found`

Que.	**7. who | sort – logfile > newfile**
Command	who | sort – logfile > newfile
Output	`ubuntu@ubuntu:~/DEPSTAR/CE$ who | sort - logfile > newfile`

CONCLUSION:

Through this practical, I learned about managing Linux processes and controlling services. I also learned about how to monitor and manage processes, control system services, and use commands like I/O redirection and piping to work more effectively in the terminal.

PROBLEMS FACED:

Practical 4

Aim: Study of Linux File System

Que.	**1. The File Hierarchy Standard (FHS) is a specification that defines the file system hierarchy of a Linux OS. Illustrate about the use of all directories under the "\" as given in the figure.**
Output	**/boot** Contains all the files required for booting the system, such as the Linux kernel, initial RAM disk (initrd), and bootloader configuration files (e.g., GRUB). It is essential for system startup. **/bin (Binaries)** Holds essential command binaries (executables) like basic commands needed by all users such as ls, cp, mv, cat, etc. These binaries are crucial during system boot or single-user mode when other parts of the filesystem (like /usr) may not be mounted yet. **/dev (Devices)** Contains special device files that represent hardware devices (such as hard drives, USBs, keyboards, etc.) and pseudo-devices (e.g., /dev/null). Devices can be interacted with as if they were files. **/etc (Configuration)** This directory contains system-wide configuration files, scripts, and settings. For example, /etc/passwd contains user account information, and /etc/fstab defines filesystem mounts. **/lib (Libraries)** Houses shared libraries needed by the binaries in /bin and /sbin. These libraries (similar to DLL files in Windows) are necessary for the system's basic functioning. **/proc (Processes)**

A virtual filesystem that provides information about system processes and other kernel-related information. Files in /proc don't actually exist on disk; they are generated by the system and represent live system information. For example, /proc/cpuinfo contains CPU details.

/root

This is the home directory of the root user (the superuser). Unlike normal users whose home directories are in /home, the root user's directory is separate, under /.

/sbin (System Binaries)

Contains essential system binaries that are typically used by the root user for system administration tasks (e.g., fsck, shutdown, mount). These commands are important for the maintenance and repair of the system.

/tmp (Temporary)

Used for storing temporary files created by programs. The files in /tmp are typically cleared out either at boot time or periodically by the system.

/usr (User Programs)

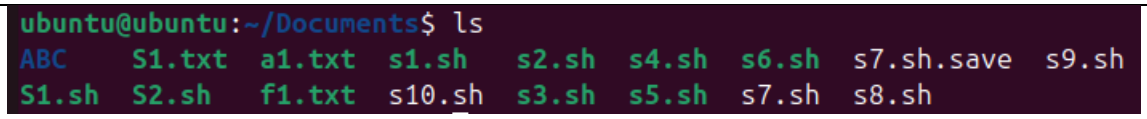
Holds user programs and utilities. It contains many important subdirectories:

a. /usr/bin: Non-essential user commands.
b. /usr/sbin: Non-essential system binaries for administrative tasks.
c. /usr/lib: Libraries for binaries in /usr/bin and /usr/sbin.
d. /usr/share: Architecture-independent files such as icons, documentation, and shared libraries.
e. /usr/local: Software and scripts that are not part of the distribution but are installed manually by the system administrator.

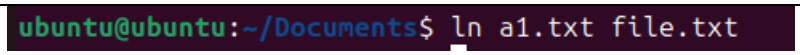
/var (Variable Files)

Stores files that are expected to change frequently. Common subdirectories include:

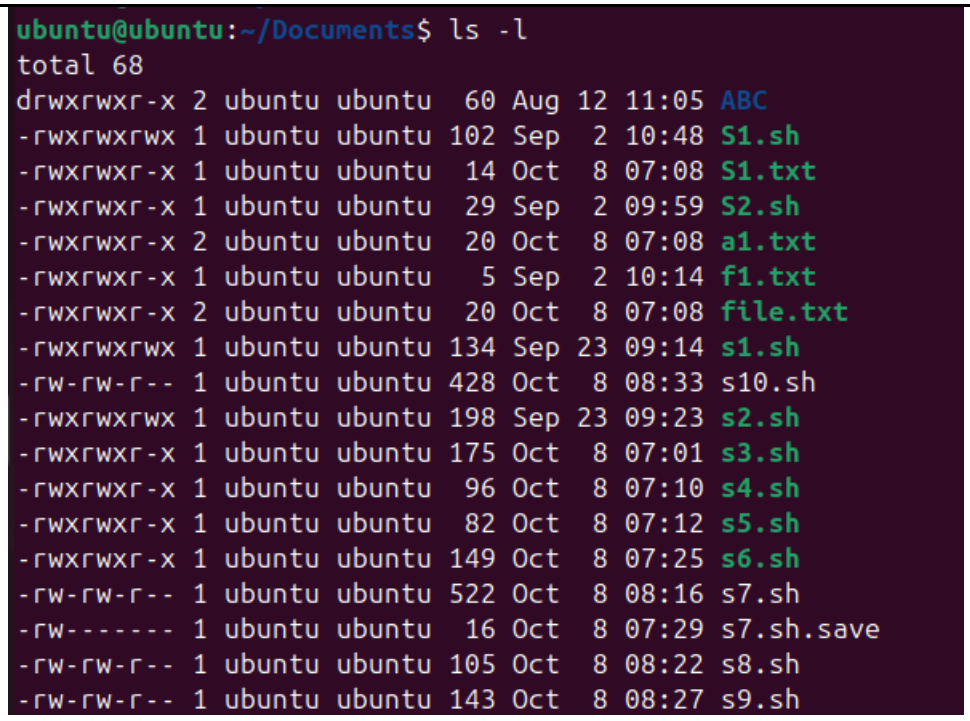
a. /var/log: System log files.
b. /var/spool: Directories for queued work, like print jobs and email.
c. /var/lib: Holds state information that programs need to preserve between reboots.

Que.	2. List out files in your directory.
Command	ls
Output	

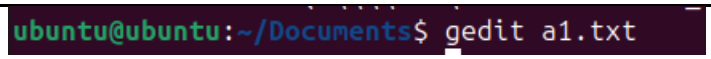
```
ubuntu@ubuntu:~/Documents$ ls
ABC     S1.txt  a1.txt  s1.sh   s2.sh  s4.sh  s6.sh  s7.sh.save  s9.sh
S1.sh   S2.sh   f1.txt  s10.sh  s3.sh  s5.sh  s7.sh  s8.sh
```

Que.	3. Create a hard link to one of the file exist in your directory. (ln [original file] [link name])
Command	ln a1.txt file.txt
Output	

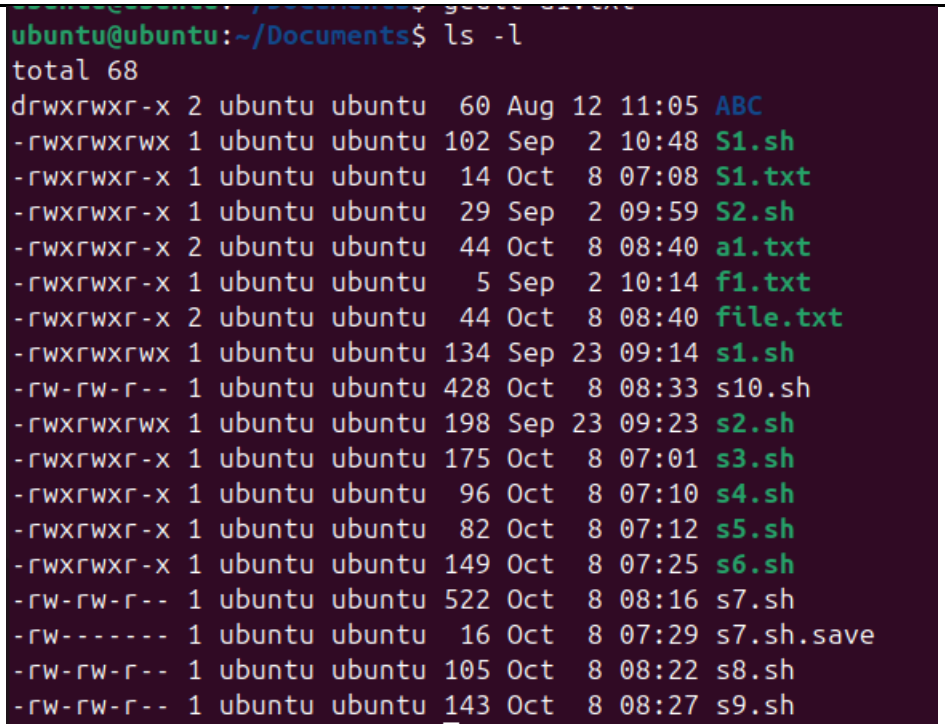
```
ubuntu@ubuntu:~/Documents$ ln a1.txt file.txt
```

Que.	4. Apply ls –l and check whether the link is created or not. Also check the size of linked file created.
Command	ls -l
Output	

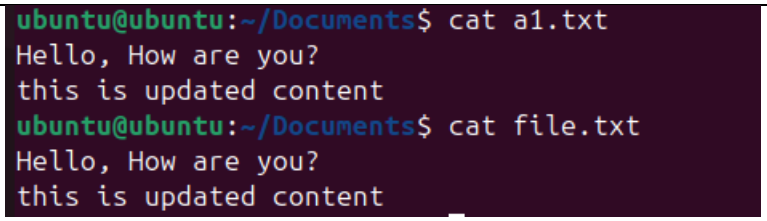
```
ubuntu@ubuntu:~/Documents$ ls -l
total 68
drwxrwxr-x 2 ubuntu ubuntu  60 Aug 12 11:05 ABC
-rwxrwxrwx 1 ubuntu ubuntu 102 Sep  2 10:48 S1.sh
-rwxrwxr-x 1 ubuntu ubuntu  14 Oct  8 07:08 S1.txt
-rwxrwxr-x 1 ubuntu ubuntu  29 Sep  2 09:59 S2.sh
-rwxrwxr-x 2 ubuntu ubuntu  20 Oct  8 07:08 a1.txt
-rwxrwxr-x 1 ubuntu ubuntu   5 Sep  2 10:14 f1.txt
-rwxrwxr-x 2 ubuntu ubuntu  20 Oct  8 07:08 file.txt
-rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh
-rw-rw-r-- 1 ubuntu ubuntu 428 Oct  8 08:33 s10.sh
-rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh
-rwxrwxr-x 1 ubuntu ubuntu 175 Oct  8 07:01 s3.sh
-rwxrwxr-x 1 ubuntu ubuntu  96 Oct  8 07:10 s4.sh
-rwxrwxr-x 1 ubuntu ubuntu  82 Oct  8 07:12 s5.sh
-rwxrwxr-x 1 ubuntu ubuntu 149 Oct  8 07:25 s6.sh
-rw-rw-r-- 1 ubuntu ubuntu 522 Oct  8 08:16 s7.sh
-rw------- 1 ubuntu ubuntu  16 Oct  8 07:29 s7.sh.save
-rw-rw-r-- 1 ubuntu ubuntu 105 Oct  8 08:22 s8.sh
-rw-rw-r-- 1 ubuntu ubuntu 143 Oct  8 08:27 s9.sh
```

Que.	5. Update the existing file.
Command	gedit a1.txt
Output	

```
ubuntu@ubuntu:~/Documents$ gedit a1.txt
```

Que.	6. Apply ls –l and check the size of linked file created.
Command	ls -l
Output	

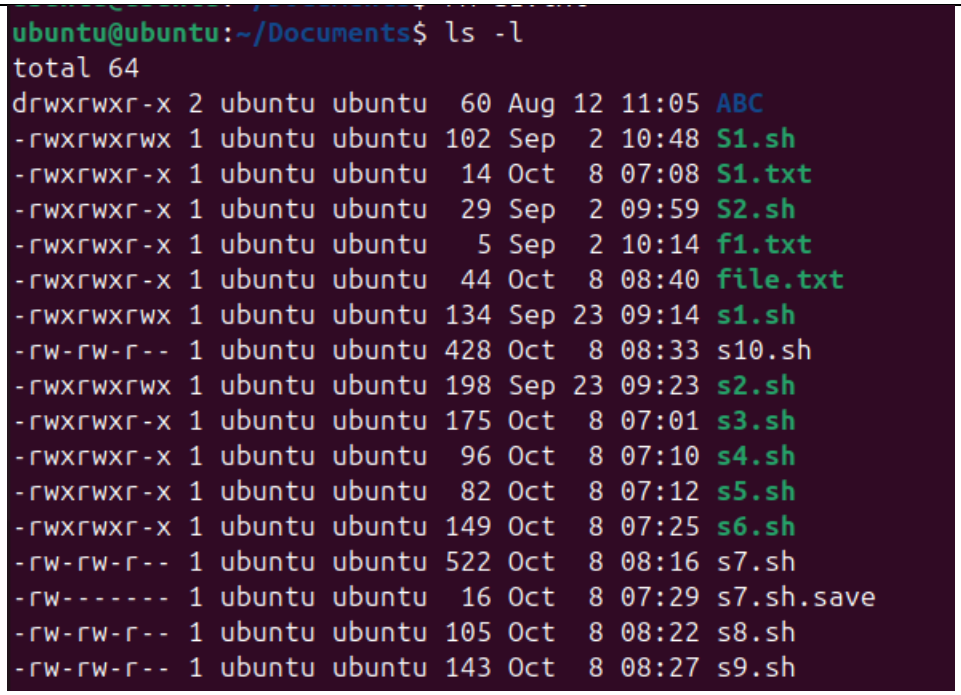
```
ubuntu@ubuntu:~/Documents$ ls -l
total 68
drwxrwxr-x 2 ubuntu ubuntu  60 Aug 12 11:05 ABC
-rwxrwxrwx 1 ubuntu ubuntu 102 Sep  2 10:48 S1.sh
-rwxrwxr-x 1 ubuntu ubuntu  14 Oct  8 07:08 S1.txt
-rwxrwxr-x 1 ubuntu ubuntu  29 Sep  2 09:59 S2.sh
-rwxrwxr-x 2 ubuntu ubuntu  44 Oct  8 08:40 a1.txt
-rwxrwxr-x 1 ubuntu ubuntu   5 Sep  2 10:14 f1.txt
-rwxrwxr-x 2 ubuntu ubuntu  44 Oct  8 08:40 file.txt
-rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh
-rw-rw-r-- 1 ubuntu ubuntu 428 Oct  8 08:33 s10.sh
-rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh
-rwxrwxr-x 1 ubuntu ubuntu 175 Oct  8 07:01 s3.sh
-rwxrwxr-x 1 ubuntu ubuntu  96 Oct  8 07:10 s4.sh
-rwxrwxr-x 1 ubuntu ubuntu  82 Oct  8 07:12 s5.sh
-rwxrwxr-x 1 ubuntu ubuntu 149 Oct  8 07:25 s6.sh
-rw-rw-r-- 1 ubuntu ubuntu 522 Oct  8 08:16 s7.sh
-rw------- 1 ubuntu ubuntu  16 Oct  8 07:29 s7.sh.save
-rw-rw-r-- 1 ubuntu ubuntu 105 Oct  8 08:22 s8.sh
-rw-rw-r-- 1 ubuntu ubuntu 143 Oct  8 08:27 s9.sh
```

Que.	7. Check the content of both the files and write your observation.
Command	cat a1.txt cat file.txt
Output	

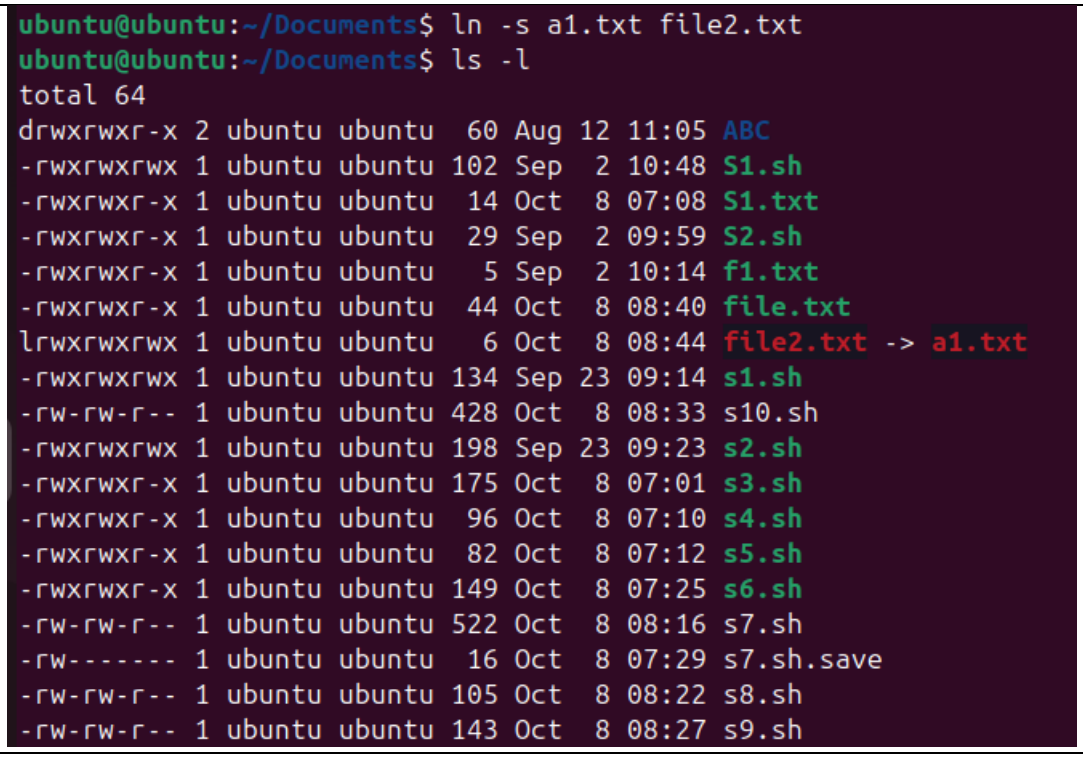
```
ubuntu@ubuntu:~/Documents$ cat a1.txt
Hello, How are you?
this is updated content
ubuntu@ubuntu:~/Documents$ cat file.txt
Hello, How are you?
this is updated content
```

Que.	8. Delete the existing file on which you have created the link.
Command	rm a1.txt
Output	

```
ubuntu@ubuntu:~/Documents$ rm a1.txt
```

Que.	9. Apply ls –l and observer the output.
Command	ls -l
Output	

```
ubuntu@ubuntu:~/Documents$ ls -l
total 64
drwxrwxr-x 2 ubuntu ubuntu  60 Aug 12 11:05 ABC
-rwxrwxrwx 1 ubuntu ubuntu 102 Sep  2 10:48 S1.sh
-rwxrwxr-x 1 ubuntu ubuntu  14 Oct  8 07:08 S1.txt
-rwxrwxr-x 1 ubuntu ubuntu  29 Sep  2 09:59 S2.sh
-rwxrwxr-x 1 ubuntu ubuntu   5 Sep  2 10:14 f1.txt
-rwxrwxr-x 1 ubuntu ubuntu  44 Oct  8 08:40 file.txt
-rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh
-rw-rw-r-- 1 ubuntu ubuntu 428 Oct  8 08:33 s10.sh
-rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh
-rwxrwxr-x 1 ubuntu ubuntu 175 Oct  8 07:01 s3.sh
-rwxrwxr-x 1 ubuntu ubuntu  96 Oct  8 07:10 s4.sh
-rwxrwxr-x 1 ubuntu ubuntu  82 Oct  8 07:12 s5.sh
-rwxrwxr-x 1 ubuntu ubuntu 149 Oct  8 07:25 s6.sh
-rw-rw-r-- 1 ubuntu ubuntu 522 Oct  8 08:16 s7.sh
-rw------- 1 ubuntu ubuntu  16 Oct  8 07:29 s7.sh.save
-rw-rw-r-- 1 ubuntu ubuntu 105 Oct  8 08:22 s8.sh
-rw-rw-r-- 1 ubuntu ubuntu 143 Oct  8 08:27 s9.sh
```

Que.	10. Perform exercise 2 to 9 for creation of soft link and write your observation. (ln –s [original file] [link name])
Command	ln -s a1.txt file2.txt ls -l
Output	

```
ubuntu@ubuntu:~/Documents$ ln -s a1.txt file2.txt
ubuntu@ubuntu:~/Documents$ ls -l
total 64
drwxrwxr-x 2 ubuntu ubuntu  60 Aug 12 11:05 ABC
-rwxrwxrwx 1 ubuntu ubuntu 102 Sep  2 10:48 S1.sh
-rwxrwxr-x 1 ubuntu ubuntu  14 Oct  8 07:08 S1.txt
-rwxrwxr-x 1 ubuntu ubuntu  29 Sep  2 09:59 S2.sh
-rwxrwxr-x 1 ubuntu ubuntu   5 Sep  2 10:14 f1.txt
-rwxrwxr-x 1 ubuntu ubuntu  44 Oct  8 08:40 file.txt
lrwxrwxrwx 1 ubuntu ubuntu   6 Oct  8 08:44 file2.txt -> a1.txt
-rwxrwxrwx 1 ubuntu ubuntu 134 Sep 23 09:14 s1.sh
-rw-rw-r-- 1 ubuntu ubuntu 428 Oct  8 08:33 s10.sh
-rwxrwxrwx 1 ubuntu ubuntu 198 Sep 23 09:23 s2.sh
-rwxrwxr-x 1 ubuntu ubuntu 175 Oct  8 07:01 s3.sh
-rwxrwxr-x 1 ubuntu ubuntu  96 Oct  8 07:10 s4.sh
-rwxrwxr-x 1 ubuntu ubuntu  82 Oct  8 07:12 s5.sh
-rwxrwxr-x 1 ubuntu ubuntu 149 Oct  8 07:25 s6.sh
-rw-rw-r-- 1 ubuntu ubuntu 522 Oct  8 08:16 s7.sh
-rw------- 1 ubuntu ubuntu  16 Oct  8 07:29 s7.sh.save
-rw-rw-r-- 1 ubuntu ubuntu 105 Oct  8 08:22 s8.sh
-rw-rw-r-- 1 ubuntu ubuntu 143 Oct  8 08:27 s9.sh
```

Que.	**11. Write difference between hard link and soft link.**
Output	Hard Link: • Points directly to the inode of a file. • Both the original file and hard link share the same inode. • Deleting the original file does not remove the hard link; data remains accessible. Soft Link (Symbolic Link): • Acts as a shortcut to another file. • Points to the file's path, not the inode. If the original file is deleted, the soft link becomes broken.

Que.	**12. Apply ls –l /dev/sda1**
Command	ls –l /dev/sda1
Output	`ubuntu@ubuntu:~/Documents$ ls -l /dev/sda` `brw-rw---- 1 root disk 8, 0 Aug 12 09:33 /dev/sda`

Que.	**13. To get an overview of local and remote file system devices and the amount of free space available, run the df command**
Command	ls -df
Output	`ubuntu@ubuntu:~/Documents$ ls -df` `.`

Que.	**14. Apply df –h and see the difference in the output**
Command	df -h

Output	`ubuntu@ubuntu:~/Documents$ df -h`
	`Filesystem Size Used Avail Use% Mounted on`
	`tmpfs 197M 1.9M 195M 1% /run`
	`/dev/sr0 5.7G 5.7G 0 100% /cdrom`
	`/cow 984M 312M 672M 32% /`
	`tmpfs 984M 8.0K 984M 1% /dev/shm`
	`tmpfs 5.0M 8.0K 5.0M 1% /run/lock`
	`tmpfs 984M 600K 984M 1% /tmp`
	`tmpfs 197M 160K 197M 1% /run/user/1000`

Que.	**15. For more detailed information about space used by a certain directory tree, use the du command. Apply du /home/ID_No**
Command	du

Output	`ubuntu@ubuntu:~/Documents$ du`
	`0 ./ABC`
	`64 .`

Que.	**16. Apply du –h /home/ID_No and see the difference in the output**
Command	Du -h /home/ubuntu
Output	

```
ubuntu@ubuntu:~/Documents$ du -h /home/ubuntu
4.0K    /home/ubuntu/Desktop
0       /home/ubuntu/.cache/gvfsd
4.0K    /home/ubuntu/.cache/mesa_shader_cache/25
8.0K    /home/ubuntu/.cache/mesa_shader_cache/3a
8.0K    /home/ubuntu/.cache/mesa_shader_cache/d2
12K     /home/ubuntu/.cache/mesa_shader_cache/d4
4.0K    /home/ubuntu/.cache/mesa_shader_cache/39
12K     /home/ubuntu/.cache/mesa_shader_cache/ed
16K     /home/ubuntu/.cache/mesa_shader_cache/a2
12K     /home/ubuntu/.cache/mesa_shader_cache/33
4.0K    /home/ubuntu/.cache/mesa_shader_cache/29
4.0K    /home/ubuntu/.cache/mesa_shader_cache/28
8.0K    /home/ubuntu/.cache/mesa_shader_cache/34
12K     /home/ubuntu/.cache/mesa_shader_cache/a9
8.0K    /home/ubuntu/.cache/mesa_shader_cache/1b
4.0K    /home/ubuntu/.cache/mesa_shader_cache/5d
4.0K    /home/ubuntu/.cache/mesa_shader_cache/b3
8.0K    /home/ubuntu/.cache/mesa_shader_cache/f2
12K     /home/ubuntu/.cache/mesa_shader_cache/09
8.0K    /home/ubuntu/.cache/mesa_shader_cache/c5
12K     /home/ubuntu/.cache/mesa_shader_cache/fa
8.0K    /home/ubuntu/.cache/mesa_shader_cache/57
8.0K    /home/ubuntu/.cache/mesa_shader_cache/c3
4.0K    /home/ubuntu/.cache/mesa_shader_cache/8c
8.0K    /home/ubuntu/.cache/mesa_shader_cache/e1
20K     /home/ubuntu/.cache/mesa_shader_cache/75
8.0K    /home/ubuntu/.cache/mesa_shader_cache/e7
```

Que.	**17. Use the lsblk command to list the details of a specified block device or all the available devices**
Command	lsblk
Output	

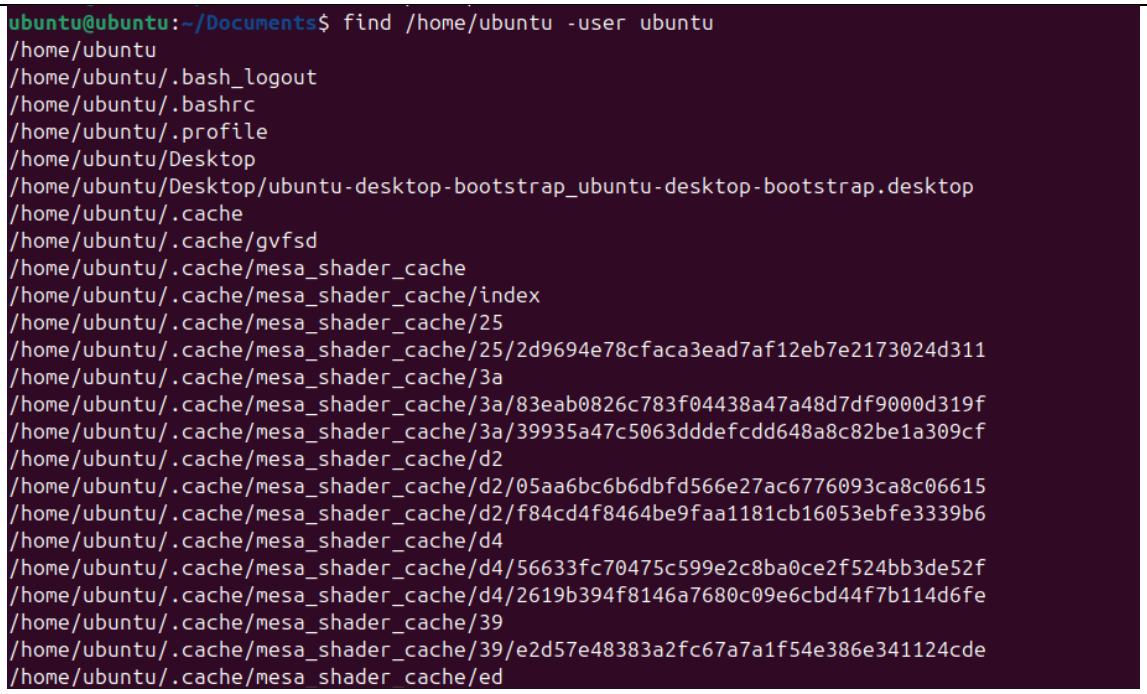
```
ubuntu@ubuntu:~/Documents$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0   1.6G  1 loop /rofs
loop1    7:1    0 457.5M  1 loop
loop2    7:2    0 868.1M  1 loop
loop3    7:3    0  10.7M  1 loop /snap/firmware-updater/127
loop4    7:4    0 269.6M  1 loop /snap/firefox/4173
loop5    7:5    0 505.1M  1 loop /snap/gnome-42-2204/176
loop6    7:6    0  38.7M  1 loop /snap/snapd/21465
loop7    7:7    0  74.2M  1 loop /snap/core22/1380
loop8    7:8    0     4K  1 loop /snap/bare/5
loop9    7:9    0   476K  1 loop /snap/snapd-desktop-integration/157
loop10   7:10   0  10.3M  1 loop /snap/snap-store/1124
loop11   7:11   0  91.7M  1 loop /snap/gtk-common-themes/1535
loop12   7:12   0 137.3M  1 loop /snap/thunderbird/470
loop13   7:13   0 116.7M  1 loop /snap/ubuntu-desktop-bootstrap/171
sda      8:0    0    25G  0 disk
sr0     11:0    1   5.7G  0 rom  /cdrom
```

Que.	18. Apply locate passwd and see the output.
Command	locate passwd
Output	ubuntu@ubuntu:~/Documents$ locate passwd /etc/passwd /etc/passwd- /etc/pam.d/chpasswd /etc/pam.d/passwd /etc/security/opasswd /rofs/etc/passwd /rofs/etc/pam.d/chpasswd /rofs/etc/pam.d/passwd /rofs/etc/security/opasswd /rofs/usr/bin/gpasswd /rofs/usr/bin/grub-mkpasswd-pbkdf2 /rofs/usr/bin/ldappasswd /rofs/usr/bin/passwd /rofs/usr/lib/python3/dist-packages/cloudinit/sources/helpers/vmware/imc/config_passwd.py /rofs/usr/lib/python3/dist-packages/cloudinit/sources/helpers/vmware/imc/__pycache__/config_passwd.cpython-312.pyc /rofs/usr/lib/tmpfiles.d/passwd.conf /rofs/usr/lib/x86_64-linux-gnu/samba/libsmbpasswdparser-samba4.so.0 /rofs/usr/sbin/chgpasswd /rofs/usr/sbin/chpasswd /rofs/usr/sbin/update-passwd /rofs/usr/share/base-passwd /rofs/usr/share/base-passwd/group.master /rofs/usr/share/base-passwd/passwd.master /rofs/usr/share/bash-completion/completions/chpasswd /rofs/usr/share/bash-completion/completions/gpasswd /rofs/usr/share/bash-completion/completions/grub-mkpasswd-pbkdf2

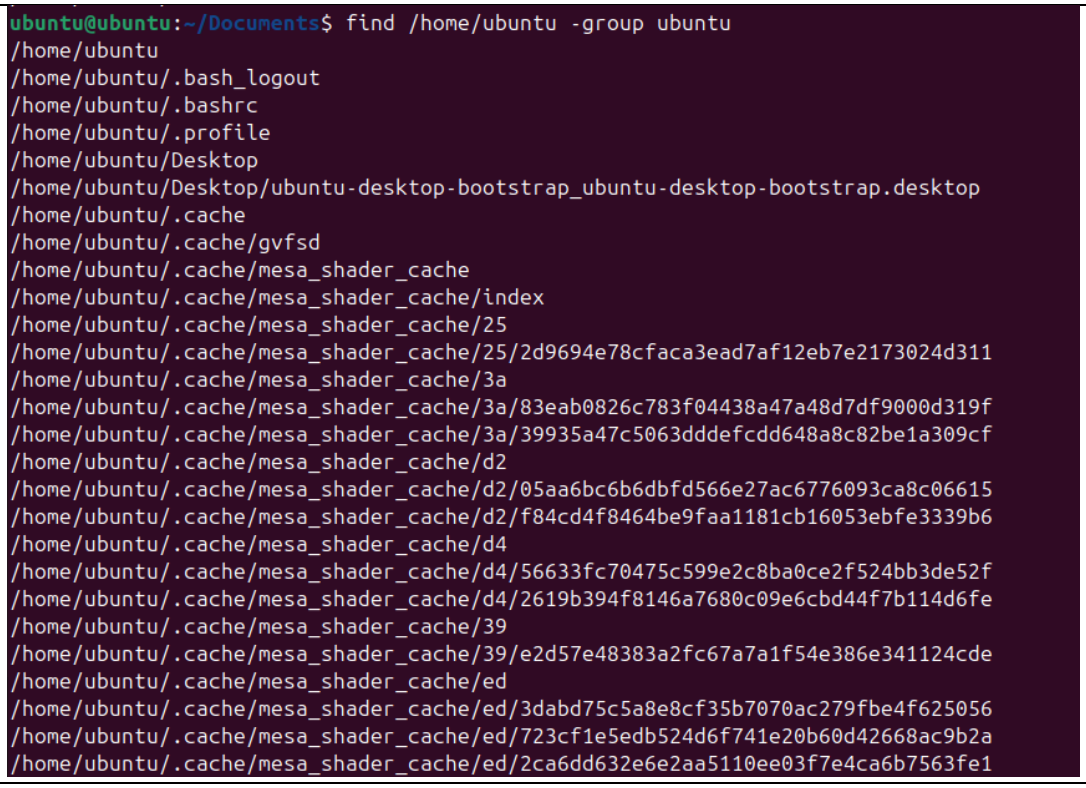
Que.	19. To search for files by file name, use the -name FILENAME option. With this option, find returns the path to files matching FILENAME exactly. Search for files named sshd_config starting from the / directory. (find / -name sshd_config)
Command	find / -name sshd_config
Output	ubuntu@ubuntu:~/Documents$ find / -name sshd_config find: ‘/etc/credstore’: Permission denied find: ‘/etc/credstore.encrypted’: Permission denied find: ‘/etc/cups/ssl’: Permission denied find: ‘/etc/polkit-1/rules.d’: Permission denied /etc/ssh/sshd_config find: ‘/etc/ssl/private’: Permission denied find: ‘/etc/sssd’: Permission denied find: ‘/etc/sudoers.d’: Permission denied find: ‘/home/installer’: Permission denied find: ‘/home/shiv’: Permission denied find: ‘/proc/tty/driver’: Permission denied find: ‘/proc/1/task/1/fd’: Permission denied find: ‘/proc/1/task/1/fdinfo’: Permission denied find: ‘/proc/1/task/1/ns’: Permission denied find: ‘/proc/1/fd’: Permission denied find: ‘/proc/1/map_files’: Permission denied find: ‘/proc/1/fdinfo’: Permission denied find: ‘/proc/1/ns’: Permission denied find: ‘/proc/2/task/2/fd’: Permission denied find: ‘/proc/2/task/2/fdinfo’: Permission denied find: ‘/proc/2/task/2/ns’: Permission denied find: ‘/proc/2/fd’: Permission denied find: ‘/proc/2/map_files’: Permission denied find: ‘/proc/2/fdinfo’: Permission denied

Que.	20. search for files starting in the / directory that end in .txt.
Command	find / -name .txt
Output	ubuntu@ubuntu:~/Documents$ find / -name .txt find: '/etc/credstore': Permission denied find: '/etc/credstore.encrypted': Permission denied find: '/etc/cups/ssl': Permission denied find: '/etc/polkit-1/rules.d': Permission denied find: '/etc/ssl/private': Permission denied find: '/etc/sssd': Permission denied find: '/etc/sudoers.d': Permission denied find: '/home/installer': Permission denied find: '/home/shiv': Permission denied find: '/proc/tty/driver': Permission denied find: '/proc/1/task/1/fd': Permission denied find: '/proc/1/task/1/fdinfo': Permission denied find: '/proc/1/task/1/ns': Permission denied find: '/proc/1/fd': Permission denied find: '/proc/1/map_files': Permission denied find: '/proc/1/fdinfo': Permission denied find: '/proc/1/ns': Permission denied find: '/proc/2/task/2/fd': Permission denied find: '/proc/2/task/2/fdinfo': Permission denied find: '/proc/2/task/2/ns': Permission denied find: '/proc/2/fd': Permission denied find: '/proc/2/map_files': Permission denied find: '/proc/2/fdinfo': Permission denied

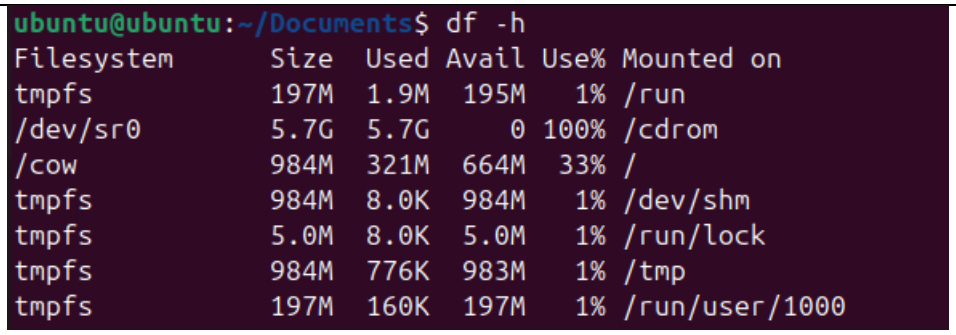
Que.	21. search for files in the /etc/ directory that contain the word, "pass".
Command	find /etc -name "pass*"
Output	ubuntu@ubuntu:~/Documents$ find /etc -name "pass*" find: '/etc/credstore': Permission denied find: '/etc/credstore.encrypted': Permission denied find: '/etc/cups/ssl': Permission denied /etc/pam.d/passwd /etc/passwd find: '/etc/polkit-1/rules.d': Permission denied find: '/etc/ssl/private': Permission denied find: '/etc/sssd': Permission denied find: '/etc/sudoers.d': Permission denied /etc/passwd-

Que.	**22. Search for files owned by user in the /home/ID_No directory.**
Command	find /home/ubuntu -user ubuntu
Output	

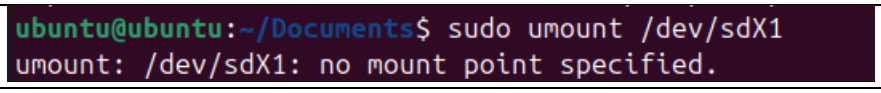
```
ubuntu@ubuntu:~/Documents$ find /home/ubuntu -user ubuntu
/home/ubuntu
/home/ubuntu/.bash_logout
/home/ubuntu/.bashrc
/home/ubuntu/.profile
/home/ubuntu/Desktop
/home/ubuntu/Desktop/ubuntu-desktop-bootstrap_ubuntu-desktop-bootstrap.desktop
/home/ubuntu/.cache
/home/ubuntu/.cache/gvfsd
/home/ubuntu/.cache/mesa_shader_cache
/home/ubuntu/.cache/mesa_shader_cache/index
/home/ubuntu/.cache/mesa_shader_cache/25
/home/ubuntu/.cache/mesa_shader_cache/25/2d9694e78cfaca3ead7af12eb7e2173024d311
/home/ubuntu/.cache/mesa_shader_cache/3a
/home/ubuntu/.cache/mesa_shader_cache/3a/83eab0826c783f04438a47a48d7df9000d319f
/home/ubuntu/.cache/mesa_shader_cache/3a/39935a47c5063dddefcdd648a8c82be1a309cf
/home/ubuntu/.cache/mesa_shader_cache/d2
/home/ubuntu/.cache/mesa_shader_cache/d2/05aa6bc6b6dbfd566e27ac6776093ca8c06615
/home/ubuntu/.cache/mesa_shader_cache/d2/f84cd4f8464be9faa1181cb16053ebfe3339b6
/home/ubuntu/.cache/mesa_shader_cache/d4
/home/ubuntu/.cache/mesa_shader_cache/d4/56633fc70475c599e2c8ba0ce2f524bb3de52f
/home/ubuntu/.cache/mesa_shader_cache/d4/2619b394f8146a7680c09e6cbd44f7b114d6fe
/home/ubuntu/.cache/mesa_shader_cache/39
/home/ubuntu/.cache/mesa_shader_cache/39/e2d57e48383a2fc67a7a1f54e386e341124cde
/home/ubuntu/.cache/mesa_shader_cache/ed
```

Que.	**23. Search for files owned by the group user in the /home/ID_No.**
Command	find /home/ubuntu -group ubuntu
Output	

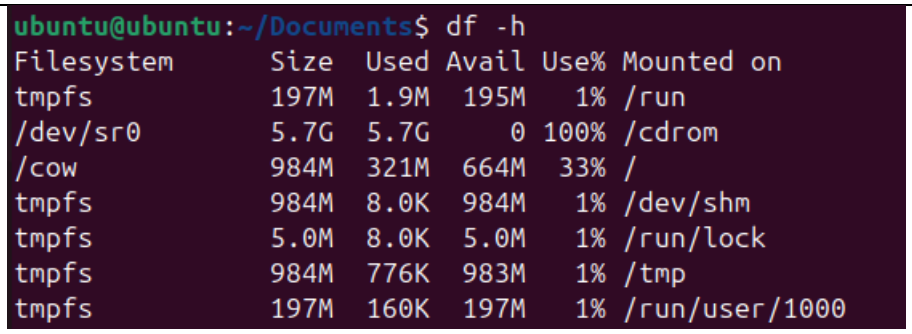
```
ubuntu@ubuntu:~/Documents$ find /home/ubuntu -group ubuntu
/home/ubuntu
/home/ubuntu/.bash_logout
/home/ubuntu/.bashrc
/home/ubuntu/.profile
/home/ubuntu/Desktop
/home/ubuntu/Desktop/ubuntu-desktop-bootstrap_ubuntu-desktop-bootstrap.desktop
/home/ubuntu/.cache
/home/ubuntu/.cache/gvfsd
/home/ubuntu/.cache/mesa_shader_cache
/home/ubuntu/.cache/mesa_shader_cache/index
/home/ubuntu/.cache/mesa_shader_cache/25
/home/ubuntu/.cache/mesa_shader_cache/25/2d9694e78cfaca3ead7af12eb7e2173024d311
/home/ubuntu/.cache/mesa_shader_cache/3a
/home/ubuntu/.cache/mesa_shader_cache/3a/83eab0826c783f04438a47a48d7df9000d319f
/home/ubuntu/.cache/mesa_shader_cache/3a/39935a47c5063dddefcdd648a8c82be1a309cf
/home/ubuntu/.cache/mesa_shader_cache/d2
/home/ubuntu/.cache/mesa_shader_cache/d2/05aa6bc6b6dbfd566e27ac6776093ca8c06615
/home/ubuntu/.cache/mesa_shader_cache/d2/f84cd4f8464be9faa1181cb16053ebfe3339b6
/home/ubuntu/.cache/mesa_shader_cache/d4
/home/ubuntu/.cache/mesa_shader_cache/d4/56633fc70475c599e2c8ba0ce2f524bb3de52f
/home/ubuntu/.cache/mesa_shader_cache/d4/2619b394f8146a7680c09e6cbd44f7b114d6fe
/home/ubuntu/.cache/mesa_shader_cache/39
/home/ubuntu/.cache/mesa_shader_cache/39/e2d57e48383a2fc67a7a1f54e386e341124cde
/home/ubuntu/.cache/mesa_shader_cache/ed
/home/ubuntu/.cache/mesa_shader_cache/ed/3dabd75c5a8e8cf35b7070ac279fbe4f625056
/home/ubuntu/.cache/mesa_shader_cache/ed/723cf1e5edb524d6f741e20b60d42668ac9b2a
/home/ubuntu/.cache/mesa_shader_cache/ed/2ca6dd632e6e2aa5110ee03f7e4ca6b7563fe1
```

Que.	**24. Apply df -h to see the partitions.**
Command	df -h
Output	

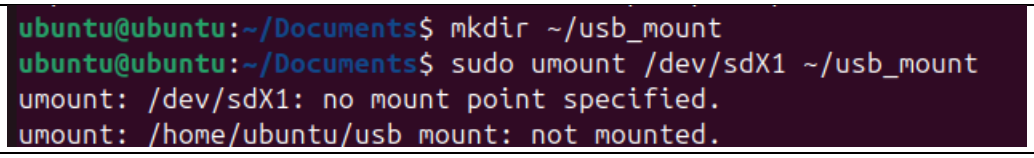
```
ubuntu@ubuntu:~/Documents$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           197M  1.9M  195M   1% /run
/dev/sr0        5.7G  5.7G     0 100% /cdrom
/cow            984M  321M  664M  33% /
tmpfs           984M  8.0K  984M   1% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
tmpfs           984M  776K  983M   1% /tmp
tmpfs           197M  160K  197M   1% /run/user/1000
```

Que.	**25. Unmount your USB drive by locating its path**
Command	sudo umount /dev/sdX1
Output	

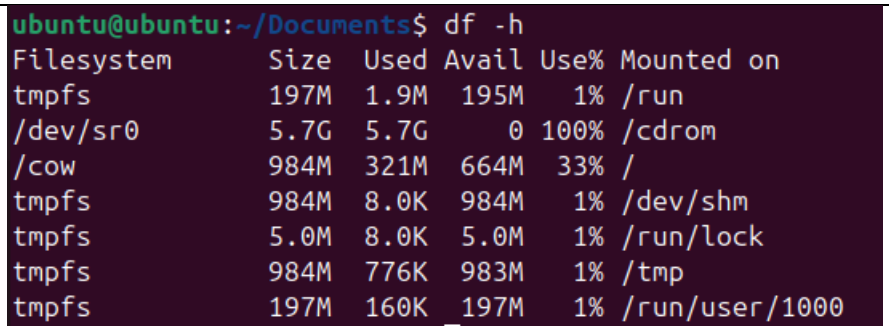
```
ubuntu@ubuntu:~/Documents$ sudo umount /dev/sdX1
umount: /dev/sdX1: no mount point specified.
```

Que.	**26. Again, apply df -h and check your USB if it is accessible or not.**
Command	df -h
Output	

```
ubuntu@ubuntu:~/Documents$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           197M  1.9M  195M   1% /run
/dev/sr0        5.7G  5.7G     0 100% /cdrom
/cow            984M  321M  664M  33% /
tmpfs           984M  8.0K  984M   1% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
tmpfs           984M  776K  983M   1% /tmp
tmpfs           197M  160K  197M   1% /run/user/1000
```

Que.	**27. Mount your USB drive into directory. (Create one directory and inside that apply mount drive directory name command)**
Command	mkdir ~/usb_mount sudo unmount /dev/sdX1 ~/usb_mount
Output	

```
ubuntu@ubuntu:~/Documents$ mkdir ~/usb_mount
ubuntu@ubuntu:~/Documents$ sudo umount /dev/sdX1 ~/usb_mount
umount: /dev/sdX1: no mount point specified.
umount: /home/ubuntu/usb_mount: not mounted.
```

Que.	28. Again, apply df -h command and check whether it is mounted or not.
Command	df -h
Output	

```
ubuntu@ubuntu:~/Documents$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           197M  1.9M  195M   1% /run
/dev/sr0        5.7G  5.7G     0 100% /cdrom
/cow            984M  321M  664M  33% /
tmpfs           984M  8.0K  984M   1% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
tmpfs           984M  776K  983M   1% /tmp
tmpfs           197M  160K  197M   1% /run/user/1000
```

Practical 5

Aim: Shell scripting

Exercise:

Que.	1. Check whether the given file exists or not.

Command	#!/bin/bash
	
	read -p "Enter the filename: " filename
	
	if [-f "$filename"]; then
	  echo "$filename exists"
	else
	  echo "$filename doesn't exist"
	fi

Output	ubuntu@ubuntu:~/Documents$ bash s1.sh Enter Filename S1.txt File exists

Que.	2. Check whether the argument passed from command line is file or directory.

Command	if test -f "$1"; then
	  echo "$1 is a file"
	elif test -d "$1"; then
	  echo "$1 is a directory"
	else
	  echo "$1 is neither file nor directory"
	fi

Output	ubuntu@ubuntu:~/Documents$ bash s2.sh s2.sh: line 1: !#/bin/bash: No such file or directory Enter File/Directory name S1.txt 'S1.txt' is a File ubuntu@ubuntu:~/Documents$ bash s2.sh s2.sh: line 1: !#/bin/bash: No such file or directory Enter File/Directory name ABC 'ABC' is a Directory

Que.	**3. List out all empty files in current working directory. Directory may contain subdirectories also.**

Command	#!/bin/bash
	
	check() {
	  for i in "$1"/*;
	  do
	  if [-f "$i"]; then
	  if [! -s "$i"]; then
	  echo "Empty file: $i"
	  fi
	  elif [-d "$i"]; then
	  check "$i"
	  fi
	  done
	}
	check .

Output	ubuntu@ubuntu:~/Documents$ bash s3.sh Empty file : ./S1.txt Empty file : ./a1.txt

Que.	**4. Give two file names as command line arguments and check both the files are same or different. If they are same then delete the second file otherwise suggest what changes are required to make 1st file similar to second file.**

Command	if cmp -s "$1" "$2"; then
	  echo "both files are same"
	else
	  echo "different"
	  diff "$1" "$2"
	fi

Output	ubuntu@ubuntu:~/Documents$ bash s4.sh S1.txt a1.txt different 1c1 < Hello World! --- > Hello, How are you?

Que.	**5. Print multiplication table of given number**
Command	echo "Enter a number: " read n for i in {1..10} do echo "$n * $i = $((n * i))" done
Output	ubuntu@ubuntu:~/Documents$ bash s5.sh Enter a Number 19 19 * 1 = 19 19 * 2 = 38 19 * 3 = 57 19 * 4 = 76 19 * 5 = 95 19 * 6 = 114 19 * 7 = 133 19 * 8 = 152 19 * 9 = 171 19 * 10 = 190

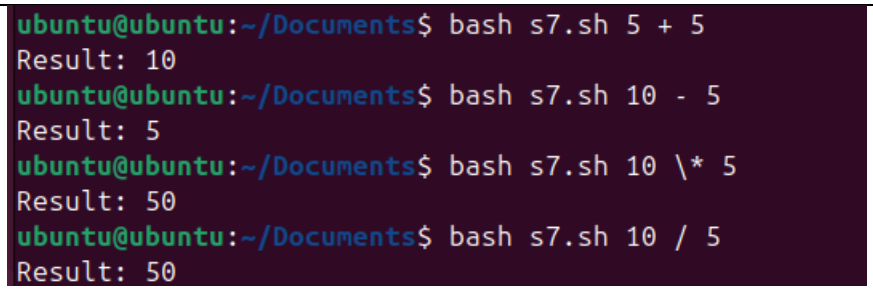
Que.	**6. Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.**

Command	for i in *
	do
	  if test -f "$i"; then
	  if test -x "$i"; then
	  echo "Executable"
	  else
	  echo "Making executable"
	  chmod +x "$i"
	  fi
	  fi
	done

Output	ubuntu@ubuntu:~/Documents$ bash s6.sh Executable Making executable Making executable Making executable Making executable Executable Executable Making executable Making executable Making executable Making executable

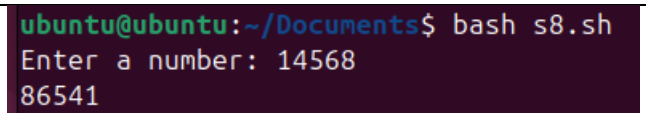
Que.	**7. Write a shell script for arithmetic calculator using command line arguments.**

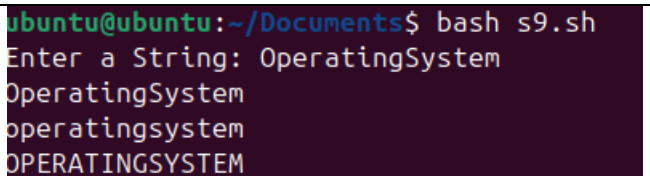
Command	if ["$#" -ne 3]; then
	  echo "Usage: $0 <number1> <operator> <number2>"
	  echo "Operators: +, -, *, /"
	  exit 1
	fi
	
	num1=$1
	operator=$2
	num2=$3
	
	case $operator in
	  +)
	  result=$(bc <<< "$num1 + $num2")
	  ;;
	  -)
	  result=$(bc <<< "$num1 - $num2")
	  ;;
	  *)
	  result=$(bc <<< "$num1 * $num2")
	  ;;
	  /)
	  if ["$num2" -eq 0]; then
	  echo "Error: Division by zero is not allowed."
	  exit 2
	  fi
	  result=$(echo "scale=2; $num1 / $num2" | bc)
	  ;;
	  *)
	  echo "Error: Invalid operator, Use +, -, *, or /."
	  exit 1
	  ;;
	esac
	
	echo "Result: $result"

Output	

Que.	**8. Write a script to print a given number in reversed order.**

Command	read -p "Enter a number: " n
	
	while [$n -gt 0]
	do
	  p=$((n % 10))
	  echo -n "$p"
	  n=$((n / 10))
	done
	
	echo

Output	

Que.	**9. Write a script to convert string from lower to upper and upper to lower case.**
Command	read -p "Enter a string: " s l=$(tr '[:upper:]' '[:lower:]' <<< "$s") u=$(tr '[:lower:]' '[:upper:]' <<< "$s") echo "$s" echo "$l" echo "$u"
Output	

Que.	**10. Shell script to Create a menu as shown below using the case statement** 1) list of files 2) today's date 3) users of system

	4) processes of user 5) exit to prompt

Command	while true; do
	  echo "1. List of files"
	  echo "2. Today's date"
	  echo "3. Users of system"
	  echo "4. Process of user"
	  echo "5. Exit to prompt"
	
	  read -p "Enter your choice: " n
	
	  case $n in
	  1)
	  echo "List of files:"
	  ls
	  ;;
	  2)
	  echo "Today's date:"
	  date
	  ;;
	  3)
	  echo "Users of system:"
	  who
	  ;;
	  4)
	  echo "Processes of current user:"
	  ps -u $(whoami)
	  ;;
	  5)
	  echo "Exit to prompt"
	  exit 0
	  ;;
	  *)
	  echo "Invalid option"
	  ;;
	  esac
	done

Output

```
ubuntu@ubuntu:~/Documents$ bash s10.sh
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 1
List of files:
ABC    S1.txt  a1.txt  s1.sh   s2.sh  s4.sh  s6.sh  s7.sh.save  s9.sh
S1.sh  S2.sh   f1.txt  s10.sh  s3.sh  s5.sh  s7.sh  s8.sh
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 2
Today's date:3
Tue Oct  8 08:33:22 UTC 2024
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 3
Users of System:
ubuntu   seat0        2024-08-12 09:32 (login screen)
ubuntu   :0           2024-08-12 09:32 (:0)
```

```
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 5
Exit to prompt
ubuntu@ubuntu:~/Documents$
```

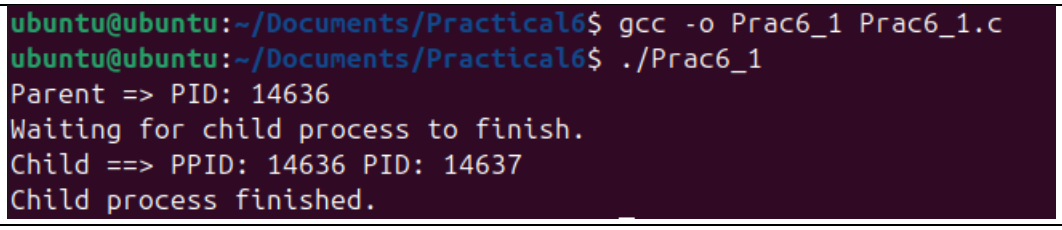
```
Enter your choice: 3
Users of System:
ubuntu   seat0        2024-08-12 09:32 (login screen)
ubuntu   :0           2024-08-12 09:32 (:0)
1. List of files
2. Today's date
3. USers of system
4. Process of user
5. Exit to prompt
Enter your choice: 4
Processes of current user:
    PID TTY          TIME CMD
   1659 ?        00:00:07 systemd
   1660 ?        00:00:00 (sd-pam)
   1677 ?        00:00:14 pipewire
   1678 ?        00:00:00 pipewire
   1680 ?        00:00:09 wireplumber
   1681 ?        00:00:06 pipewire-pulse
   1682 ?        00:00:01 gnome-keyring-d
   1687 ?        00:00:13 dbus-daemon
   1720 tty2     00:00:00 gdm-x-session
   1724 tty2     00:13:06 Xorg
   1923 tty2     00:00:00 gnome-session-b
   2112 ?        00:00:00 at-spi-bus-laun
   2119 ?        00:00:01 dbus-daemon
   2170 ?        00:00:00 gcr-ssh-agent
   2171 ?        00:00:00 gnome-session-c
   2187 ?        00:00:00 gvfsd
   2199 ?        00:00:00 gvfsd-fuse
   2225 ?        00:00:00 gnome-session-b
   2254 ?        00:18:42 gnome-shell
```

Practical 6

Aim: Process control system calls:

Que.	**The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)**

Code	#include <stdio.h>
	#include <stdlib.h>
	#include <sys/types.h>
	#include <sys/wait.h>
	#include <unistd.h>
	
	int main(void) {
	  pid_t pid = fork();
	  if (pid == 0) {
	  printf("Child ==> PPID: %d PID: %d\n", getppid(), getpid());
	  } else if (pid > 0) {
	  printf("Parent => PID: %d\n", getpid());
	  printf("Waiting for child process to finish.\n");
	  wait(NULL);
	  printf("Child process finished.\n");
	  } else {
	  printf("Unable to create child process.\n");
	  }
	  return EXIT_SUCCESS;
	}

Output	

Que.	**The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)**

Code	#include <stdio.h>
	#include <sys/types.h>
	#include < unistd.h>
	
	int main() {
	  fork();
	  printf("Hello\n");
	  return 0;
	}

Output	ubuntu@ubuntu:~/Documents/Practical6$ gcc -o Prac6_2 Prac6_2.c ubuntu@ubuntu:~/Documents/Practical6$ ./Prac6_2 Hello Hello

Que.	**The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)**

Code	#include <stdio.h>
	#include <sys/types.h>
	#include < unistd.h>
	
	int main() {
	  fork();
	  fork();
	  fork();
	  printf("Hello\n");
	  return 0;
	}

Output	ubuntu@ubuntu:~/Documents/Practical6$ gcc -o Prac6_3 Prac6_3.c ubuntu@ubuntu:~/Documents/Practical6$ ./Prac6_3 Hello Hello Hello ubuntu@ubuntu:~/Documents/Practical6$ Hello Hello Hello Hello Hello

Que.	**The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)**

Code	#include <stdio.h>
	#include <sys/types.h>
	#include <unistd.h>
	
	void forkexample() {
	  if (fork() == 0) {
	  printf("Child!\n");
	  } else {
	  printf("Parent!\n");
	  }
	}
	
	int main() {
	  forkexample();
	  return 0;
	}

Output	ubuntu@ubuntu:~/Documents/Practical6$ gcc -o Prac6_4 Prac6_4.c ubuntu@ubuntu:~/Documents/Practical6$ ./Prac6_4 Parent! Child!

Summary:

In this practical exercise, the goal was to demonstrate the use of the fork() system call in process creation and management. The specific aim was to create a user-defined number of processes, noting that the fork() system call inherently creates processes in powers of 2.

Summary of Exercises:

1. In this exercise, a simple program is created using the fork() system call to demonstrate the creation of a child process from a parent process. The child process prints its process ID (PID) and the parent's PID (PPID), while the parent process waits for the child to complete before exiting. This shows the synchronization between parent and child processes using wait().

2. This exercise involved a program that uses fork() to create a new process, which then prints "Hello world!" from both the parent and the child process. The program demonstrates that fork() creates an exact copy of the calling process, leading to multiple outputs due to multiple process instances.

3. In this example, three consecutive calls to fork() are made, resulting in the creation of multiple processes. The number of processes created is in the power of 2, specifically 2^3=8. Each process prints "Hello", demonstrating the exponential growth in the number of processes created when fork() is called multiple times.

4. Here, the fork() system call is encapsulated within a function. Depending on whether fork() returns 0 (child process) or a non-zero value (parent process), the function prints either "Child!" or "Parent!". This example shows how fork() can be used within a function to manage process creation and differentiate behavior between parent and child processes.

Conclusion:

The practical session illustrated the mechanics of the fork() system call and its impact on process creation. By experimenting with different uses of fork(), such as creating multiple processes, synchronizing parent and child processes, and managing process behaviors, you gained a deeper understanding of how process control works in UNIX-like operating systems. The exercises demonstrated that each call to fork() results in a new child process, with the total number of processes doubling with each call, hence following the power of 2 growth pattern.

Practical 7

Aim: Demonstration of execve () and wait () system calls along with zombie and orphan states.

Que.	1. Zombie Process

Code	#include <stdlib.h>
	#include <sys/types.h>
	#include <unistd.h>
	
	int main() {
	  pid_t child_pid = fork();
	
	  if (child_pid > 0) {
	  sleep(50);
	  } else {
	  exit(0);
	  }
	
	  return 0;
	}

Output	ubuntu@ubuntu:~/Documents/Practical7$ g++ Prac7_1.c -o Prac7_1 ubuntu@ubuntu:~/Documents/Practical7$ ./Prac7_1

Que.	2. Orphan Process

Code	#include <stdio.h>
	#include <sys/types.h>
	#include <unistd.h>
	
	int main() {
	  int pid = fork();
	
	  if (pid > 0) {
	  printf("in parent process\n");
	  } else if (pid == 0) {
	  sleep(30);
	  printf("in child process\n");
	  }
	
	  return 0;
	}

Output	ubuntu@ubuntu:~/Documents/Practical7$ g++ Prac7_2.c -o Prac7_2 ubuntu@ubuntu:~/Documents/Practical7$ ./Prac7_2 in parent process ubuntu@ubuntu:~/Documents/Practical7$ in child process

Que.	3. Execvparent Execvchild

Code	**Execvparent**
	#include <stdio.h>
	#include <unistd.h>
	#include <stdlib.h>
	
	int main(int argc, char *argv[]) {
	  printf("\n-> PID of parent.c = %d ", getpid());
	
	  int parent = fork();
	
	  if (parent == -1) {
	  printf("\n-> Some errors in calling");
	  }
	
	  if (parent == 0) {
	  printf("\n-> Now execv will call child.c from child process.");
	  printf("\n-> The childe process in running.");
	
	  char *args[] = {"239", NULL};
	  execv("./execvchild", args);
	  } else {

```
        printf("\n-> Now parent is running \n");
    }

    return 0;
}
```

Execvchild

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    printf("\n-> Now we are in child.c");
    printf("\n-> The PID of child.c = %d", getpid());
    printf("\n-> %s", *argv);

    return 0;
}
```

Output

```
ubuntu@ubuntu:~/Documents/Practical7$ g++ Prac7_3_2.c -o Prac7_3_2
ubuntu@ubuntu:~/Documents/Practical7$ ./Prac7_3_2

-> Now we are in child.c
-> The PID of child.c = 16692
-> ./Prac7_3_2ubuntu@ubuntu:~/Documents/Practical7$ ^C
```

Summary:

The objective of Practical 7 was to demonstrate the use of the execv() and wait() system calls in the context of process control, and to illustrate the creation and handling of zombie and orphan processes.

Summary of Exercises:

1. **Zombie Process:**
    - A zombie process is a process that has completed execution (terminated) but still has an entry in the process table. This happens when the parent process does not call wait() to read the child's exit status.
    - In this exercise, the program forks to create a child process, which immediately terminates using exit(0), while the parent process sleeps for 50 seconds. During this sleep period, the child becomes a zombie because the parent does not call wait() to clear the terminated child's entry from the process table.

2. **Orphan Process:**
    - An orphan process is a child process whose parent has terminated or exited. Orphan processes are adopted by the init process (PID 1) in UNIX-based systems.
    - In this example, a child process is created using fork(). The parent process prints "in parent process" and then terminates. Meanwhile, the child process sleeps for 30 seconds before

printing "in child process". When the child process finally wakes up, its parent is no longer active, making it an orphan, and it is then adopted by the init process.

3. **execv() System Call:**

- The execv() system call replaces the current process image with a new process image. It is commonly used to run a different program within the same process.
- This exercise involves two files: execvparent.c and execvchild.c.
  - execvparent.c: The parent process is created, which in turn forks a child process. The child process uses the execv() call to replace its image with that of execvchild.c, effectively running the new program (execvchild.c).
  - execvchild.c: This program prints the PID of the current child process and some arguments passed by the parent. This exercise demonstrates how the execv() call transfers control to another program and how data can be passed between them.

Conclusion:

Practical 7 provided hands-on experience with advanced process control techniques in UNIX-like systems. The exercises helped demonstrate how to manage processes, replace them using the execv() system call, and handle special states like zombie and orphan processes. This understanding is fundamental in system programming, where process management is critical to ensuring resource efficiency and stability of applications.

Practical 8

Aim: Implementation of Process Scheduling Algorithm:

Que.	A. FCFS

Code	#include <iostream>
	#include <algorithm>
	using namespace std;
	typedef struct {
	int arrival_time, burst_time, waiting_time, pid, TAT;
	} Process;
	bool compare(Process a, Process b) {
	return a.arrival_time < b.arrival_time;
	}
	int main()
	{
	int N, total_waiting_time = 0, start_time = 0;
	float average_waiting_time = 0;
	cout << "Enter Number of Processes : ";
	cin >> N;
	cout << endl;
	Process process[N];
	for(int i=0; i<N; i++) {
	cout << "Enter Burst time of P[" << (i+1) << "] : ";
	cin >> process[i].burst_time;
	cout << "Enter Arrival time of P[" << (i+1) << "] : ";
	cin >> process[i].arrival_time;
	process[i].pid = (i+1);
	cout << endl;
	}
	sort(process, process+N, compare);
	cout << "PID Arrival time Burst time Waiting Time TAT" << endl;
	for(int i=0; i<N; i++) {
	if(i == 0) {
	process[i].waiting_time = 0;
	start_time = process[i].arrival_time;
	}
	else
	process[i].waiting_time = start_time - process[i].arrival_time;

```
total_waiting_time += process[i].waiting_time;
 start_time += process[i].burst_time;
 process[i].TAT = start_time - process[i].arrival_time;
 cout << process[i].pid << "\t " << process[i].arrival_time << "\t\t" <<
process[i].burst_time << "\t " << process[i].waiting_time << "\t " << process[i].TAT <<
endl;
 }
 average_waiting_time = (float)total_waiting_time/N;
 cout << "\nTotal Waiting time : " << total_waiting_time << endl;
 cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
```

Output

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_1.c -o Prac8_1
ubuntu@ubuntu:~/Documents$ ./Prac8_1
Enter Number of Processes : 5

Enter Burst time of P[1] : 6
Enter Arrival time of P[1] : 2

Enter Burst time of P[2] : 2
Enter Arrival time of P[2] : 5

Enter Burst time of P[3] : 8
Enter Arrival time of P[3] : 1

Enter Burst time of P[4] : 3
Enter Arrival time of P[4] : 0

Enter Burst time of P[5] : 4
Enter Arrival time of P[5] : 4

PID Arrival time Burst time Waiting Time TAT
4        0              3        0       3
3        1              8        2       10
1        2              6        9       15
5        4              4        13      17
2        5              2        16      18

Total Waiting time : 40
Average Waiting time : 8
```

Que.	B. Round Robing

Code	#include <iostream>
	#include <algorithm>
	using namespace std;
	typedef struct {
	int arrival_time, burst_time, waiting_time, pid, TAT, remaining_time, completion_time;
	bool completed;
	} Process;
	bool compareArrival(Process a, Process b) {
	return (a.arrival_time < b.arrival_time);
	}
	int main()
	{
	int N, quantum, completed = 0, total_waiting_time = 0, time = 0;
	float average_waiting_time = 0;
	cout << "Enter Number of Processes : ";
	cin >> N;
	cout << "Enter Quantum Time : ";
	cin >> quantum;
	cout << endl;
	Process process[N];
	for(int i=0; i<N; i++) {
	cout << "Enter Burst time of P[" << (i+1) << "] : ";
	cin >> process[i].burst_time;
	cout << "Enter Arrival time of P[" << (i+1) << "] : ";
	cin >> process[i].arrival_time;
	process[i].pid = (i+1);
	process[i].remaining_time = process[i].burst_time;
	process[i].completed = false;
	cout << endl;
	}
	sort(process, process+N, compareArrival);
	cout << "PID Arrival time Burst time Waiting Time TAT Completion time" << endl;
	time += process[0].arrival_time;
	while(completed < N) {
	for(int i=0; i<N; i++) {
	if(process[i].completed == true) continue;
	else if(process[i].arrival_time > time) break;
	else {
	if(process[i].remaining_time > quantum) {
	time += quantum;
	process[i].remaining_time -= quantum;
	}
	else {
	time += process[i].remaining_time;
	process[i].remaining_time = 0;
	process[i].completed = true;

```
 process[i].completion_time = time;
 process[i].TAT = process[i].completion_time - process[i].arrival_time;
 process[i].waiting_time = process[i].TAT - process[i].burst_time;
 total_waiting_time += process[i].waiting_time;
 completed++;
 cout << process[i].pid << "\t " << process[i].arrival_time << "\t\t" <<
process[i].burst_time << "\t " << process[i].waiting_time << "\t " << process[i].TAT <<
"\t\t" << process[i].completion_time << endl;
 }
 }
 }
 }
 average_waiting_time = (float)total_waiting_time/N;
 cout << "\nTotal Waiting time : " << total_waiting_time << endl;
 cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
```

Output

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_2.c -o Prac8_2
ubuntu@ubuntu:~/Documents$ ./Prac8_2
Enter Number of Processes : 3
Enter Quantum Time : 2

Enter Burst time of P[1] : 4
Enter Arrival time of P[1] : 0

Enter Burst time of P[2] : 3
Enter Arrival time of P[2] : 0

Enter Burst time of P[3] : 5
Enter Arrival time of P[3] : 0

PID Arrival time Burst time Waiting Time TAT Completion time
1        0              4        4       8              8
2        0              3        6       9              9
3        0              5        7       12             12

Total Waiting time : 17
Average Waiting time : 5.66667
```

Que.	C. SJF

Code	#include <iostream>
	#include <algorithm>
	using namespace std;
	typedef struct {
	int arrival_time, burst_time, waiting_time, pid, TAT;
	} Process;
	bool compareArrival(Process a, Process b) {
	return (a.arrival_time < b.arrival_time);
	}
	bool compareBurst(Process a, Process b) {
	return (a.burst_time < b.burst_time);
	}
	void sortByBurst(Process process[], int N);
	int main()
	{
	int N, total_waiting_time = 0, start_time = 0;
	float average_waiting_time = 0;
	cout << "Enter Number of Processes : ";
	cin >> N;
	cout << endl;
	Process process[N];
	for(int i=0; i<N; i++) {
	cout << "Enter Burst time of P[" << (i+1) << "] : ";
	cin >> process[i].burst_time;
	cout << "Enter Arrival time of P[" << (i+1) << "] : ";
	cin >> process[i].arrival_time;
	process[i].pid = (i+1);
	cout << endl;
	}
	sort(process, process+N, compareArrival);
	sortByBurst(process, N);
	cout << "PID Arrival time Burst time Waiting Time TAT" << endl;
	start_time = process[0].arrival_time;
	for(int i=0; i<N; i++) {
	if(i == 0)
	process[i].waiting_time = 0;
	else
	process[i].waiting_time = start_time - process[i].arrival_time;
	total_waiting_time += process[i].waiting_time;
	start_time += process[i].burst_time;
	process[i].TAT = start_time - process[i].arrival_time;
	cout << process[i].pid << "\t " << process[i].arrival_time << "\t\t" <<
	process[i].burst_time << "\t " << process[i].waiting_time << "\t " << process[i].TAT <<
	endl;
	}
	average_waiting_time = (float)total_waiting_time/N;

```
 cout << "\nTotal Waiting time : " << total_waiting_time << endl;
 cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
void sortByBurst(Process process[], int N) {
 int start_time = process[0].arrival_time;
 int j;
 for(int i=0; i<N-1; i++) {
 for(j=i+1; process[j].arrival_time <= process[i].burst_time+start_time && j<N; j++);
 if(process[i].arrival_time != process[i+1].arrival_time)
 sort(process+i+1, process+j, compareBurst);
 else
 sort(process+i, process+j, compareBurst);
 start_time += process[i].burst_time;
 }
}
```

Output

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_3.c -o Prac8_3
ubuntu@ubuntu:~/Documents$ ./Prac8_3
Enter Number of Processes : 5

Enter Burst time of P[1] : 6
Enter Arrival time of P[1] : 2

Enter Burst time of P[2] : 2
Enter Arrival time of P[2] : 5

Enter Burst time of P[3] : 8
Enter Arrival time of P[3] : 1

Enter Burst time of P[4] : 3
Enter Arrival time of P[4] : 0

Enter Burst time of P[5] : 4
Enter Arrival time of P[5] : 4

PID Arrival time Burst time Waiting Time TAT
4        0              3        0       3
1        2              6        1       7
2        5              2        4       6
5        4              4        7       11
3        1              8        14      22

Total Waiting time : 26
Average Waiting time : 5.2
```

Que.	D. Priority Scheduling
Code	**Non-Preemptive**

```
#include <iostream>
#include <algorithm>
using namespace std;
typedef struct {
 int arrival_time, burst_time, waiting_time, pid, priority, TAT;
} Process;
bool compareArrival(Process a, Process b) {
 return (a.arrival_time < b.arrival_time);
}
bool comparePriority(Process a, Process b) {
 return (a.priority > b.priority);
}
void sortByPriority(Process process[], int N);
int main()
{
 int N, total_waiting_time = 0, start_time = 0;
 float average_waiting_time = 0;
 cout << "Enter Number of Processes : ";
 cin >> N;
 cout << endl;
 Process process[N];
 for(int i=0; i<N; i++) {
 cout << "Enter Burst time of P[" << (i+1) << "] : ";
 cin >> process[i].burst_time;
 cout << "Enter Arrival time of P[" << (i+1) << "] : ";
 cin >> process[i].arrival_time;
 cout << "Enter Priority level of P[" << (i+1) << "] : ";
 cin >> process[i].priority;
 process[i].pid = (i+1);
 cout << endl;
 }
 sort(process, process+N, compareArrival);
 sortByPriority(process, N);
cout << "PID Priority Arrival time Burst time Waiting Time TAT" << endl;
 start_time = process[0].arrival_time;
 for(int i=0; i<N; i++) {
 if(i == 0)
 process[i].waiting_time = 0;
 else
 process[i].waiting_time = start_time - process[i].arrival_time;
 total_waiting_time += process[i].waiting_time;
 start_time += process[i].burst_time;
 process[i].TAT = start_time - process[i].arrival_time;
 cout << process[i].pid << "\t" << process[i].priority << "\t " << process[i].arrival_time
<< "\t\t " << process[i].burst_time << "\t " << process[i].waiting_time << "\t\t" <<
```

```
process[i].TAT << endl;
 }
 average_waiting_time = (float)total_waiting_time/N;
 cout << "\nTotal Waiting time : " << total_waiting_time << endl;
 cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
void sortByPriority(Process process[], int N) {
 int start_time = process[0].arrival_time;
 int j;
 for(int i=0; i<N-1; i++) {
 for(j=i+1; process[j].arrival_time <= process[i].burst_time+start_time && j<N; j++);
 if(process[i].arrival_time != process[i+1].arrival_time)
 sort(process+i+1, process+j, comparePriority);
 else
 sort(process+i, process+j, comparePriority);
 start_time += process[i].burst_time;
 }
}
```

Preemptive

```
#include <iostream>
#include <algorithm>
using namespace std;
typedef struct {
 int arrival_time, burst_time, waiting_time, pid, priority, TAT, remaining_time,
completion_time;
 bool completed;
} Process;
bool compareArrival(Process a, Process b) {
 return (a.arrival_time < b.arrival_time);
}
int main()
{
 int N, total_waiting_time = 0, time = 0, completed = 0;
 float average_waiting_time = 0;
 cout << "Enter Number of Processes : ";
 cin >> N;
 cout << endl;
 Process process[N];
 for(int i=0; i<N; i++) {
 cout << "Enter Burst time of P[" << (i+1) << "] : ";
 cin >> process[i].burst_time;
 cout << "Enter Arrival time of P[" << (i+1) << "] : ";
 cin >> process[i].arrival_time;
 cout << "Enter Priority level of P[" << (i+1) << "] : ";
 cin >> process[i].priority;
```

```
 process[i].completed = false;
 process[i].remaining_time = process[i].burst_time;
 process[i].pid = (i+1);
 cout << endl;
 }
 sort(process, process+N, compareArrival);
 time += process[0].arrival_time;
 cout << "PID Priority Arrival time Burst time Waiting Time TAT Completion time" <<
endl;
 while(completed < N) {
 int max_priority = -1;
 int id = -1;
 for(int i=0; i<N; i++) {
 if(process[i].completed == true) continue;
 else if(process[i].arrival_time > time) break;
 else {
 if(process[i].priority > max_priority) {
 max_priority = process[i].priority;
 id = i;
 }
 }
 }
 time++;
 process[id].remaining_time--;
 if(process[id].remaining_time == 0) {
process[id].completed = true;
 process[id].completion_time = time;
 process[id].TAT = process[id].completion_time - process[id].arrival_time;
 process[id].waiting_time = process[id].TAT - process[id].burst_time;
 total_waiting_time += process[id].waiting_time;
 completed++;
 cout << process[id].pid << "\t" << process[id].priority << "\t " <<
process[id].arrival_time << "\t\t" << process[id].burst_time << "\t " <<
process[id].waiting_time << "\t " << process[id].TAT << "\t\t" <<
process[id].completion_time << endl;
 }
 }
 average_waiting_time = (float)total_waiting_time/N;
 cout << "\nTotal Waiting time : " << total_waiting_time << endl;
 cout << "Average Waiting time : " << average_waiting_time << endl;
return 0;
}
```

Output	**Non-Preemptive**

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_4_1.c -o Prac8_4_1
ubuntu@ubuntu:~/Documents$ ./Prac8_4_1
Enter Number of Processes : 5

Enter Burst time of P[1] : 4
Enter Arrival time of P[1] : 0
Enter Priority level of P[1] : 3

Enter Burst time of P[2] : 3
Enter Arrival time of P[2] : 0
Enter Priority level of P[2] : 2

Enter Burst time of P[3] : 7
Enter Arrival time of P[3] : 6
Enter Priority level of P[3] : 3

Enter Burst time of P[4] : 4
Enter Arrival time of P[4] : 11
Enter Priority level of P[4] : 1

Enter Burst time of P[5] : 2
Enter Arrival time of P[5] : 12
Enter Priority level of P[5] : 2
```

```
PID Priority Arrival time Burst time Waiting Time TAT
1       3        0                4       0              4
2       2        0                3       4              7
3       3        6                7       1              8
5       2        12               2       2              4
4       1        11               4       5              9

Total Waiting time : 12
Average Waiting time : 2.4
```

Preemptive

```
ubuntu@ubuntu:~/Documents$ g++ Prac8_4_2.c -o Prac8_4_2
ubuntu@ubuntu:~/Documents$ ./Prac8_4_2
Enter Number of Processes : 5

Enter Burst time of P[1] : 4
Enter Arrival time of P[1] : 0
Enter Priority level of P[1] : 3

Enter Burst time of P[2] : 3
Enter Arrival time of P[2] : 0
Enter Priority level of P[2] : 2

Enter Burst time of P[3] : 7
Enter Arrival time of P[3] : 6
Enter Priority level of P[3] : 3

Enter Burst time of P[4] : 4
Enter Arrival time of P[4] : 11
Enter Priority level of P[4] : 1

Enter Burst time of P[5] : 2
Enter Arrival time of P[5] : 12
Enter Priority level of P[5] : 2
```

```
PID Priority Arrival time Burst time Waiting Time TAT Completion time
1       3        0             4         0         4              4
3       3        6             7         0         7              13
2       2        0             3         11        14             14
5       2        12            2         2         4              16
4       1        11            4         5         9              20

Total Waiting time : 18
Average Waiting time : 3.6
```

Practical 9

Aim: Find out the case where threads are helpful.

A) With the help of posix API showcase the power of threads. Compare the Execution of single Process with threads execution.

B) Perform Thread synchronization using counting semaphores and mutual exclusion using mutex

Case Where Threads Are Helpful:

Threads are useful when you want to perform multiple tasks concurrently, especially for applications that require parallel processing, such as:

- **Web Servers**: Handling multiple client requests simultaneously.
- **Scientific Computations**: Performing large computations concurrently to leverage multi-core systems.
- **Real-Time Systems**: Managing tasks in real time, like controlling sensors in IoT devices.
- **Multimedia Applications**: Processing multiple audio or video streams concurrently.

Threads are lightweight and can be more efficient than creating new processes because they share the same memory space and resources of the parent process.

Que.	A. With the help of posix API showcase the power of threads. Compare the Execution of single Process with threads execution.

Code	**1. Single Process Execution (Without Threads)**
	#include <stdio.h>
	#include <unistd.h>
	#include <time.h>
	
	void task1() {
	for (int i = 0; i < 5; i++) {
	printf("Task 1 - Iteration %d\n", i);
	sleep(1);
	}
	}
	
	void task2() {
	for (int i = 0; i < 5; i++) {
	printf("Task 2 - Iteration %d\n", i);
	sleep(1);
	}

```
    }

    int main() {
        clock_t start = clock();
        task1();
        task2();
        clock_t end = clock();

        double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("Execution Time (Single Process): %.2f seconds\n", time_taken);

        return 0;
    }
```

2. Multi-threaded Execution (Using POSIX Threads)

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <time.h>

void* task1(void* arg) {
   for (int i = 0; i < 5; i++) {
      printf("Task 1 (Thread) - Iteration %d\n", i);
      sleep(1);
   }
   return NULL;
}

void* task2(void* arg) {
   for (int i = 0; i < 5; i++) {
      printf("Task 2 (Thread) - Iteration %d\n", i);
      sleep(1);
   }
   return NULL;
}

int main() {
   pthread_t t1, t2;
   clock_t start = clock();

   pthread_create(&t1, NULL, task1, NULL);
   pthread_create(&t2, NULL, task2, NULL);

   pthread_join(t1, NULL);
   pthread_join(t2, NULL);

   clock_t end = clock();
```

```
    double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Execution Time (Multi-threaded): %.2f seconds\n", time_taken);

    return 0;
}
```

Output

```
ubuntu@ubuntu:~/Documents/Practical9$ gedit Prac9_1_1.c
ubuntu@ubuntu:~/Documents/Practical9$ g++ Prac9_1_1.c -o Prac9_1_1
ubuntu@ubuntu:~/Documents/Practical9$ ./Prac9_1_1
Task 1 - Iteration 0
Task 1 - Iteration 1
Task 1 - Iteration 2
Task 1 - Iteration 3
Task 1 - Iteration 4
Task 2 - Iteration 0
Task 2 - Iteration 1
Task 2 - Iteration 2
Task 2 - Iteration 3
Task 2 - Iteration 4
Execution Time (Single Process): 0.00 seconds
ubuntu@ubuntu:~/Documents/Practical9$ gedit Prac9_1_2.c
ubuntu@ubuntu:~/Documents/Practical9$ g++ Prac9_1_2.c -o Prac9_1_2
ubuntu@ubuntu:~/Documents/Practical9$ ./Prac9_1_2
Task 1 (Thread) - Iteration 0
Task 2 (Thread) - Iteration 0
Task 1 (Thread) - Iteration 1
Task 2 (Thread) - Iteration 1
Task 1 (Thread) - Iteration 2
Task 2 (Thread) - Iteration 2
Task 1 (Thread) - Iteration 3
Task 2 (Thread) - Iteration 3
Task 2 (Thread) - Iteration 4
Task 1 (Thread) - Iteration 4
Execution Time (Multi-threaded): 0.00 seconds
```

Que.	B. Perform Thread synchronization using counting semaphores and mutual exclusion using mutex
Code	**1. Thread Synchronization with Semaphores**:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdint.h>

#define NUM_THREADS 5

void* task(void* arg) {
   int thread_num = (intptr_t)arg;

   printf("Thread %d is running\n", thread_num);
   sleep(1);
   printf("Thread %d is done\n", thread_num);

   pthread_exit(NULL);
}

int main() {
   pthread_t threads[NUM_THREADS];
   int rc;
   intptr_t t;

   for (t = 0; t < NUM_THREADS; t++) {
      printf("Creating thread %ld\n", t);

      rc = pthread_create(&threads[t], NULL, task, (void*)t);
      if (rc) {
         printf("ERROR; return code from pthread_create() is %d\n", rc);
         exit(-1);
      }
   }

   for (t = 0; t < NUM_THREADS; t++) {
      pthread_join(threads[t], NULL);
   }

   printf("All threads completed.\n");
   return 0;
}
```

2. Mutual Exclusion with Mutex:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdint.h>

#define NUM_THREADS 5

pthread_mutex_t mutex;

void* task(void* arg) {
   int thread_num = (intptr_t)arg;

   pthread_mutex_lock(&mutex);

   printf("Thread %d is running\n", thread_num);
   sleep(1);
   printf("Thread %d is done\n", thread_num);

   pthread_mutex_unlock(&mutex);

   pthread_exit(NULL);
}

int main() {
   pthread_t threads[NUM_THREADS];
   int rc;
   intptr_t t;

   pthread_mutex_init(&mutex, NULL);

   for (t = 0; t < NUM_THREADS; t++) {
      printf("Creating thread %ld\n", t);

      rc = pthread_create(&threads[t], NULL, task, (void*)t);
      if (rc) {
         printf("ERROR; return code from pthread_create() is %d\n", rc);
         exit(-1);
      }
   }

   for (t = 0; t < NUM_THREADS; t++) {
      pthread_join(threads[t], NULL);
   }

   pthread_mutex_destroy(&mutex);
```

```
    printf("All threads completed.\n");
    return 0;
}
```

Output

1. Thread Synchronization with Semaphores:

```
ubuntu@ubuntu:~/Documents/Practical9$ g++ Prac9_2_1.c -o Prac9_2_1 -lpthread
ubuntu@ubuntu:~/Documents/Practical9$ ./Prac9_2_1
Creating thread 0
Creating thread 1
Thread 0 is running
Thread 1 is running
Creating thread 2
Thread 2 is running
Creating thread 3
Creating thread 4
Thread 4 is running
Thread 3 is running
Thread 0 is done
Thread 1 is done
Thread 2 is done
Thread 3 is done
Thread 4 is done
All threads completed.
```

2. Mutual Exclusion with Mutex:

```
ubuntu@ubuntu:~/Documents/Practical9$ g++ Prac9_2_2.c -o Prac9_2_2 -lpthread
ubuntu@ubuntu:~/Documents/Practical9$ ./Prac9_2_2
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3
Thread 0 is running
Creating thread 4
Thread 0 is done
Thread 1 is running
Thread 1 is done
Thread 2 is running
Thread 2 is done
Thread 3 is running
Thread 3 is done
Thread 4 is running
Thread 4 is done
All threads completed.
```

Summary:

This practical exercise focused on implementing thread synchronization mechanisms using mutexes and semaphores in a multi-threaded environment.

Exercises Overview:

1. **Power of Threads using POSIX API:**
    - Created multiple threads with pthread_create().
    - Compared single-process execution with multi-threaded execution, demonstrating improved performance due to parallelism.

2. **Thread Synchronization with Mutexes:**
    - Implemented a critical section accessed by multiple threads.
    - Used mutexes (pthread_mutex_t) to ensure mutual exclusion, preventing race conditions during access to shared resources.
3. **Synchronization using Counting Semaphores:**
    - Demonstrated the use of semaphores (sem_t) to manage access to a limited resource among threads.
    - Used functions like sem_init(), sem_wait(), and sem_post() to control concurrency and prevent resource contention.

Conclusion:

Practical 9 highlighted the efficiency of multi-threaded execution over single-process execution, emphasizing the role of threads in improving performance through concurrency. Synchronization techniques like mutexes and semaphores were essential in managing shared resources safely, ensuring mutual exclusion, and preventing race conditions. Threads are particularly beneficial in scenarios requiring concurrent execution, such as CPU-bound and I/O-bound tasks.

Practical 10

Aim: Implement inter process communication (IPC) using PIPEs and FIFOs.

Que.	Implement inter process communication (IPC) using PIPEs
Code	

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main() {
    int pipefds[2];
    pid_t pid;
    char write_msg[] = "Hello from parent!";
    char read_msg[50];

    // Create pipe
    if (pipe(pipefds) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid > 0) {
        close(pipefds[0]);
        write(pipefds[1], write_msg, strlen(write_msg) + 1);
        close(pipefds[1]);
    } else {
        close(pipefds[1]);
        read(pipefds[0], read_msg, sizeof(read_msg));
        printf("Child received: %s\n", read_msg);
        close(pipefds[0]);
    }
```

```
    return 0;
}
```

Output

```
ubuntu@ubuntu:~/Documents/Practical10$ g++ Prac10_1.c -o Prac10_1
ubuntu@ubuntu:~/Documents/Practical10$ ./Prac10_1
Child received: Hello from parent!
```

Que.	**Implement inter process communication (IPC) using FIFOs.**

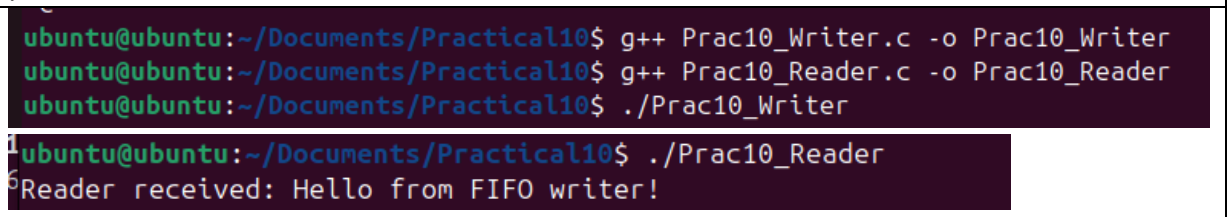
Code

Writer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

#define FIFO_FILE "/tmp/my_fifo"

int main() {
    int fd;
    char write_msg[] = "Hello from FIFO writer!";

    mkfifo(FIFO_FILE, 0666);

    fd = open(FIFO_FILE, O_WRONLY);
    write(fd, write_msg, strlen(write_msg) + 1);
    close(fd);

    return 0;
}
```

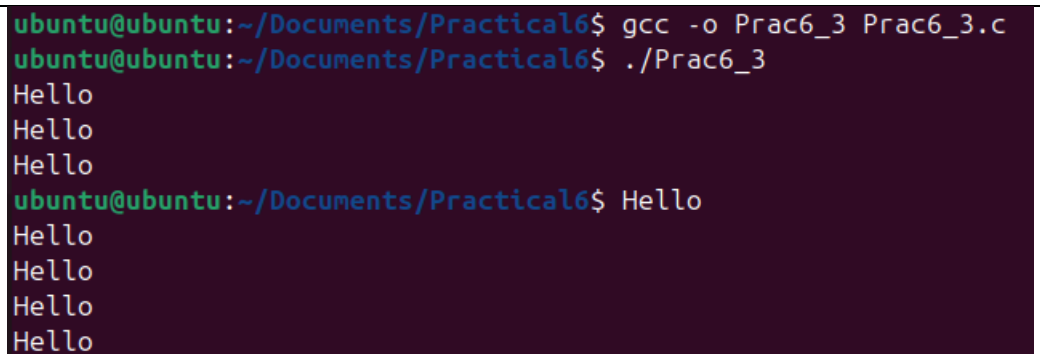
Reader

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

#define FIFO_FILE "/tmp/my_fifo"

int main() {
    int fd;
    char read_msg[50];

    fd = open(FIFO_FILE, O_RDONLY);
    read(fd, read_msg, sizeof(read_msg));
    printf("Reader received: %s\n", read_msg);
    close(fd);

    return 0;
```

	}
Output	

Que.	**The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)**

Code	#include <stdio.h>
	#include <sys/types.h>
	#include < unistd.h>
	
	int main() {
	  fork();
	  fork();
	  fork();
	  printf("Hello\n");
	  return 0;
	}

Output	

Que.	**The demonstration of fork () system call. Write a program which takes number of processes from the user and create those processes. (The fork system call will create processes in power of 2.)**

Code	#include <stdio.h>
	#include <sys/types.h>
	#include <unistd.h>
	
	void forkexample() {
	  if (fork() == 0) {
	  printf("Child!\n");

	} else {
	printf("Parent!\n");
	}
	}
	
	int main() {
	forkexample();
	return 0;
	}

Output	ubuntu@ubuntu:~/Documents/Practical6$ gcc -o Prac6_4 Prac6_4.c ubuntu@ubuntu:~/Documents/Practical6$ ./Prac6_4 Parent! Child!

Summary:

This practical exercise focused on using Inter-Process Communication (IPC) with FIFO (named pipes) and unnamed pipes to enable communication and synchronization between processes.

Exercises Overview:

1. **FIFO (Named Pipes):**
    - Created a FIFO using mkfifo() to enable communication between two unrelated processes: a writer and a reader.
    - The writer sends a message through the FIFO, and the reader waits for and receives the message. This demonstrated how blocking I/O works to synchronize the processes.
2. **Unnamed Pipes:**
    - Used the pipe() system call for communication between a parent and child process created with fork().
    - The parent writes a message to the pipe, and the child reads it, showing how pipes can facilitate data exchange between related processes.
3. **Synchronization with Threads:**
    - Demonstrated thread synchronization using mutexes and semaphores with POSIX threads.
    - Threads shared a critical section of code, where mutexes ensured mutual exclusion, and semaphores regulated access to shared resources, preventing race conditions.

Conclusion:

Practical 10 showcased the use of IPC via FIFO and pipes for communication between processes and emphasized the role of thread synchronization using mutexes and semaphores to manage shared resources and avoid conflicts in multi-threaded environments.